

Universidad ORT Uruguay

Facultad de Ingeniería

Nexden: Nilus Expert Dispatch Management System (Sistema experto de ruteos para
distribución de alimentos.)

Entregado como requisito para la obtención del título de Máster en Big Data

Ignacio Carbonell - 83652

Marcelo Espinosa - 81643

Víctor Trocki - 72865

Tutor: Alejandro Bianchi

2021

Declaración de Autoría

Nosotros, Ignacio Carbonell, Marcelo Espinosa, Víctor Trocki, declaramos que el trabajo que se presenta en esta obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos la carrera de Master en Big Data;
- Cuando hemos consultado el trabajo realizado por otros lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.



Ignacio Carbonell

25/10/2021



Marcelo Espinosa

25/10/2021



Víctor Trocki

25/10/2021

Dedicatoria

Este proyecto va dedicado a nuestras familias, que durante todo el período de cursos, trabajos y reuniones han tenido la paciencia de lidiar con nuestra ausencia, a veces frustración y siempre han sido un apoyo y motivación para hacernos avanzar.

A nuestros docentes que con una paciencia infinita nos han abierto las puertas de un universo fascinante e infinito.

Agradecimientos

A Nilus Global S.R.L. con domicilio en: Fray Justo Sta. María de Oro 2116, C1425 FOD, Buenos Aires, Argentina. www.Nilus.online

A Ady Beitler Cofundador de Nilus Global S.R.L.

Quienes compartieron con nuestro equipo todos sus datos e información a los efectos de facilitarnos el trabajo.

A nuestro tutor Alejandro Bianchi.

A los docentes consultados Juan Gabito y Nicolás Fornaro

A Sergio Yovine quien nos recibió y apoyó durante toda la carrera, nos ayudó mucho en la elección del proyecto, tutor y consultantes.

A todo el equipo docente que de forma más que generosa se han esforzado por transferir al máximo sus conocimientos.

Abstract

El presente trabajo está enfocado con el objetivo de generar una prueba de concepto para una solución de servicios tecnológicos basados en conocimiento y explotación de datos, utilizando herramientas, tácticas y estrategias de las tecnologías de la información vistas en el postgrado de la Maestría en Big Data cursado por los autores.

Dadas las necesidades de una empresa real dedicada a la distribución de alimentos en zonas carenciadas y en un entorno de actualidad innovador y de pleno crecimiento se eligió para una de las tareas definidas como relevantes, (la asignación y despacho de rutas para la distribución), un problema de gran importancia como lo es el de la selección de la/s mejor/es rutas para determinado despacho, en determinada ciudad y en determinado momento.

Si bien existen en la actualidad servicios de geolocalización que son utilizados por la empresa, los mismos carecen de información importante a la hora de elegir la mejor ruta al momento de hacer una entrega, sobretodo cuando hablamos de zonas carenciadas o alejadas de los centros más poblados y donde no hay o no existe mucha información de la periferia.

Luego de un análisis de los datos de los que dispone la empresa se seleccionó un subconjunto de los mismos con aquellos que resultaron más relevantes. A partir de los mismos y utilizando técnicas, conocimientos y tecnologías, se elaboró un conjunto de herramientas que permiten agregar a la información de los servicios existentes “una capa” de conocimiento capaz de capturar aquella información definida como relevante para el caso de negocio con el objetivo de poder clasificarlas como mejores o peores rutas en consideración de lo que resulta relevante para el negocio (distancia, estado, composición, restricciones, etc.).

Se realizó como parte del trabajo, un análisis detallado de las fuentes de datos proporcionados por la empresa de forma de tener una idea clara del grado de profundidad y calidad que se puede obtener de la implementación de una solución de estas características en el momento actual y futuro. Se realizaron también recomendaciones pertinentes que a juicio de los autores se deberían tomar si se quisiera profundizar en las soluciones presentadas en este trabajo sobre todo pensando en los beneficios de utilizar tecnologías de información predictivas y de aprendizaje automático.

Palabras clave

Nilus: empresa de triple impacto dedicada a la recuperación y distribución de alimentos para la que se hizo el siguiente trabajo.[1]

Nexden: Nilus expert dispatch management system, sistema experto de ruteo para distribución de alimentos

Neo4j: Base de datos orientada a grafos[2]

ArcGis: Algoritmo para la búsqueda de rutas en Open Street Maps[3]

OSM: Open Street Maps[4]

MongoDB: Base de datos orientada a documentos[5]

Python: Lenguaje de script[6]

GCP: Google Cloud Platform[7]

Ruteo: Acción de sugerir el camino o ruta

Last Mile: Término muy utilizado en la industria del software para señalar los programas que solucionan las rutas de distribución

Sistema experto de ruteo para distribución

Historial de Revisiones

Fecha	Responsable	Observación
31/09/2020	NC- VT – ME	Versión inicial
06/10/2020	NC- VT – ME	Versión corregida
20/10/2020	ME	Versión corregida
14/12/2020	ME	ETL en proceso
16/01/2021	NC	ETL en proceso
24/01/2021	VT	Neo4j – Análisis
07/02/2021	NC	MongoDb
10/02/2021	NC, VT	MongoDb-Neo4j int.
15/02/2021	NC – VT- ME	Diseño Arquitectura y Middleware(py)
18/02/2021	NC	impl Middleware
20/02/2021	ME	On cloud services UP
25/02/2021	NC- VT- ME	Diseño Arquitectura y Middleware(py)-ev2

28/02/2021	ME	Puesta a punto pruebas arquitectura en nube
03/03/2021 – 26/3/2021	NC- VT- ME	Ajuste general de contenidos



“En Italia se censuran con acrimonia (como hacían los Griegos) los errores notables de los edificios. Puso cierto Arquitecto Veneciano una bella fila de columnas en parage que de nada servía, y nada sostenían. Cierta mañana amaneció en ellas un Pasquín que les preguntaba: Messer colonne, che fatte quá?. A qué respondían: Non lo sappiamo in verità”
[8]

Índice

1	Introducción	14
1.1	Propósito y Alcance	14
1.2	Empresa	14
1.3	Escenario	15
1.4	Problemática y Necesidades	16
2	Visión de la solución	18
2.1	Visión general	18
2.2	Visión particular	18
3	Requerimientos	20
3.1	Vista de Casos de Uso	20
3.2	Vista de Datos	20
3.3	Consideraciones y Decisiones Tomadas	22
3.4	Cronograma	26
4	Arquitectura de la solución	27
5	Tecnologías seleccionadas	28
5.1	Descripción general	28
5.2	Base de Datos Documental	31
5.3	Middleware	32
5.4	Interfaces	33
	□ BiqQuery	33

□	MongoDB	33
□	Neo4j	33
□	OpenStreetMap	33
6	Conclusiones	34
7	Recomendaciones	36
□	Calidad, integridad y accesibilidad de los Datos	36
□	Selección de Tecnologías	36
□	Flexibilidad en implementación	36
□	“Time to Market” con visión de mediano y largo plazo	36
□	Mejora continua como estrategia transversal	37
8	Lecciones aprendidas	38
□	Entorno empresarial	38
□	Ecosistema tecnológico	38
□	Planes, Ideas y realidad	39
□	Los grupos de trabajo y el factor humano	39
9	Referencias bibliográficas	40
10	ANEXO I - Persistencia de datos en MongoDB	42
□	Creación de Base de Datos Documental en MongoDB	42
□	Carga y Exploración de Datos	43
□	Deliveries	43
□	Transporte	45

□ Viajes (Trips)	45
□ Observaciones finales	48
11 ANEXO II - Exploración de estrategia en Neo4j, Consideraciones preliminares y Análisis y selección de Algoritmos	49
Manejo de los grafos para Algoritmos:	50
Clasificación de Algoritmos:	50
o Algoritmos de Centralidad:	50
o Algoritmos de detección de comunidades.	51
o Algoritmos de Similitud	52
o Algoritmos para selección de caminos.	53
o Algoritmos de predicción de enlaces	54
o Algoritmos de inserción de nodos.	55
o Modelos de aprendizaje automático	55
o Procedimientos auxiliares.	56
12 ANEXO III - Middleware en Python	57
MongoDB a Neo - Transformar y Exportar los Datos	57
Rutas en OpenStreetMap	62
Recuperación y marcado de rutas	62
Librerías	63
Conexión a Servicios	64
Recuperación de Rutas de Neo4j - Trazado de Ruta Simple	65
Rutas Complejas	71

	Buscar si existe ruta	75
	Alta de Incidentes	77
13	ANEXO IV - Análisis detallado de datos	79
	Tabla deliveries	87
	Tabla Milestones	95
	Tabla Vehicles	99
	Tabla Verifications	103
	Tabla Trips	105

1 Introducción

1.1 Propósito y Alcance

El presente trabajo tiene como propósito documentar la implementación de una solución de tecnología de la Información en una empresa real, utilizando un conjunto de herramientas, ideas y conceptos vistos por los autores tanto en el diploma como en el Master de Big Data. Dicho trabajo se realizará conforme con los requisitos establecidos por UNIVERSIDAD ORT URUGUAY para ser presentado como trabajo final en pos de la obtención del título final.

El alcance se enmarcará dentro de la exposición de la problemática de una empresa real y la visión de la solución tecnológica a llevar adelante con el objetivo de documentar la creación de una “prueba de concepto” que demuestre la viabilidad técnica y utilidad de la solución propuesta para resolver un problema concreto.

1.2 Empresa

Nilus - Tecnología al rescate de alimentos [1]

Nilus es una empresa cuya misión es crear mercados de alimentos asequibles y nutritivos para personas de bajos ingresos con la visión de liderar la transformación digital de la industria del rescate de alimentos.



Conforma un canal de distribución de alimentos destinado a personas de bajos ingresos ofreciendo alimentos a precios sociales. Debido al uso de tecnología de rescate en tiempo real puede acceder a productos frescos que han salido del circuito comercial por estar próximos a vencer o tener defectos menores de estética o packaging. Además, brinda a sus clientes la posibilidad de realizar compras conjuntas, negociando los mejores precios y consolidando las entregas logísticas, reduciendo así los costos para los comedores sociales.

En un momento de fuerte expansión se han desarrollado una serie de herramientas tecnológicas que permiten poner en contacto aquellos que ofrecen la mercadería con los clientes, pudiendo ver stocks online y manejo de órdenes y su gestión.

En el caso concreto de la ciudad de Rosario, Argentina, donde vamos a implementar la prueba de concepto, la distribución de alimentos siempre funciona a través de un centro de acopio y distribución de alimentos. Por lo que el despacho es siempre desde el centro de acopio hacia el cliente final o Kitchen Soup (Merenderos o comedores comunitarios).

Es importante destacar también el gran interés y la total disposición de la empresa y de todo su equipo técnico, según las propias manifestaciones de su CEO, para con el presente proyecto, ya que obtener una mejora sustancial en este punto para su negocio resulta de importancia vital y estratégica de acuerdo con los planes de desarrollo y expansión actuales y futuros que se han trazado.

1.3 Escenario

El centro del negocio es el rescate de alimentos que están estresados en su vida útil pero que se encuentran en buenas condiciones sanitarias para ser consumidos. Son obtenidos principalmente a partir de donaciones de grandes cadenas de supermercados, compras directas a fabricantes y productores, y se distribuyen en comunidades de bajos ingresos.



Buenos Aires, Sao Paulo, Ciudad de México, Puerto Rico, entre otros, son los principales territorios donde la empresa busca posicionarse como la organización líder en materia de tecnología para la reducción de pérdidas y desperdicios de alimentos de América Latina y el Caribe.

1.4 Problemática y Necesidades

Si bien actualmente con el uso de las herramientas actuales la empresa puede gestionar los viajes a realizar, la logística en sí de la distribución aún no se encuentra automatizada. Se necesita entonces mejorar algunos aspectos de la distribución física de los alimentos, entre ellos un diseño más eficiente del proceso de distribución y de la planificación de las rutas para las entregas.

Es en este particular donde se encuentran dos puntos clave a resolver y donde creemos que podemos aportar valor:

- Las zonas de distribución (villas, favelas o barrios marginales), que recorren no suelen estar bien cartografiadas por los sistemas habituales (ej.: Google Maps[9]).
- Dichas zonas suelen presentar problemas de accesibilidad de diversa índole dependiendo de muchos factores (peligrosidad de determinados horarios, cortes por protestas, etc.)

A su vez, les sería de mucha ayuda poder identificar posibles puntos notables de entrega (ej.: Iglesias, Clubes Sociales, plaza pública etc.), que sean sugeridos por el sistema en contingencias no previstas como puede ser un corte de caminos u otro caso.

La solución a desarrollar debería entonces permitir modelar rutas que contemplen estas peculiaridades marcadas, que puedan sugerir lugares de entrega en contingencias no deseadas, así como aprender de las mismas a través del tiempo. Cabe aclarar que la empresa puso a disposición datos para poder comenzar a trabajar, a lo cual habría que analizar si integrar otras fuentes de datos que puedan ser relevantes como por ejemplo “escuchar” redes sociales, datos del tiempo, policía, plataformas de tránsito (Waze, Google Maps), una App desarrollada para los propios choferes, etc.

En cuanto a la obtención de los datos la idea es que para enmarcar el trabajo académico dentro de una prueba de concepto, deberíamos delimitar una zona o área geográfica, (la empresa propuso comenzar con zonas marginales de alguna provincia del interior de Argentina donde ellos estén operando como es el caso de la ciudad de Rosario), obtener de la empresa para esa zona los puntos de recogida y distribución ya existentes como para comenzar a realizar una carga inicial con datos “conocidos” y luego ir alimentando con datos que vayan surgiendo

de relevamiento realizado mediante documentación existente de vistas cartografiadas de la zona o relevamiento realizado con drones y/o datos provistos por terceros (empresas existentes, policía, emergencias móviles, etc.). La empresa sugirió utilizar los mapas de OpenStreetMap[4] en lugar de los de Google debido a que por experiencia propia los primeros son mejores sobretodo cuando se quiere acceder a zonas marginales además de que OpenStreetMap es de código abierto y de acceso público con todas las posibilidades que ello conlleva.

Otro punto importante y que es parte del desafío del presente trabajo en lo que refiere a la obtención y actualización de los datos consiste en tratar de ver el mejor modo de capturar el conocimiento que se tiene por parte de uno de los principales actores del sistema que son los choferes, conocimiento de las rutas en lo que refiere a calidad de los caminos, recorridos más cortos, horarios seguros para determinadas zonas y zonas peligrosas que se deberían evitar de forma permanente o en horarios particulares.

Sobre lo antedicho un punto a analizar para sugerir, como posibilidad futura pero que su implementación no forma parte del presente trabajo, sería proveer una aplicación o herramienta con la capacidad de poder capturar de los mismos texto y/o audio con alertas y recomendaciones que vengan de los propios choferes que puedan ser un complemento para enriquecer al sistema de recomendaciones así como implementar una pasarela de datos para capturar tanto estos como de fuentes externas como ya se mencionó. De poder hacerse viable la funcionalidad descrita, podría extenderse luego, fomentando con algún tipo de incentivo claro está, a otros actores colaboradores, “gente de a pie”, pueden ser los propios “receptores” quienes colaboren en la actualización de los datos para mantener seguras las rutas que hacen posible que los alimentos lleguen en tiempo y forma para su propio beneficio.

En otro orden entendemos que para realizar el presente trabajo desarrollando una prueba de concepto que integre, personas, procesos y tecnologías será necesario presentar los resultados de forma gráfica e intuitiva, pero en la instancia de este trabajo separadas del resto de las soluciones y herramientas con la que hoy cuenta la empresa delimitando también así el alcance y la complejidad.

2 Visión de la solución

2.1 Visión general

neXDen (nilus eXpert Dispatch engine). Sistema experto de ruteos para distribución de alimentos

La visión general de la solución consiste en la generación de un sistema experto en el despacho de ruteos para la distribución de los alimentos que maneja y administra la empresa. Mediante el análisis y procesamiento inteligente de los datos de los puntos de distribución, así como de sus restricciones (horarios, zonas, estado de la caminería, etc.) permitir la selección y propuesta de las mejores alternativas en lo que a elección de rutas se refiere para una distribución particular en un día y momento dados.

La idea de fondo es intentar aplicar a la problemática que se quiere abordar algún conjunto de las técnicas y herramientas vistas durante el transcurso del diploma y el Master en Big Data con el objetivo de dotar al sistema de capacidades mejoradas debido a la utilización "inteligente" de los datos generados por el mismo, por los que provengan del entorno donde este se desempeñe y la dinámica de la interacción entre estos. La gran capacidad de proceso y almacenamiento disponible y creciente de hoy día nos permite utilizar toda la información de la que se dispone para hacer que el sistema realice, prediga y pueda adquirir capacidades aumentadas de "aprendizaje", para casi todo aquello que consideremos relevante de acuerdo a los objetivos que se han definido para el mismo.

A lo antedicho debemos sumar que todo el desarrollo de la Analítica de Big-Data ha posibilitado la creación de un Stack tecnológico que obliga a repensar todo el desarrollo tradicional de las herramientas de software desde las interfaces, las bases de datos y el procesamiento para toma de decisiones en tiempo real.

2.2 Visión particular

Para el presente trabajo, debido a dos variables fundamentales como lo son por un lado el tiempo disponible y el estado actual general de la calidad y cantidad de los datos de los que dispone la empresa se optó por la creación de una prueba de concepto con un conjunto mínimo acotado de los datos existentes. Dicha prueba consistirá en que partiendo de los datos de punto de partida y puntos de entrega (pick-ups, y drop-offs), generar un conjunto de rutas posibles

priorizadas por algún atributo relevante (distancia, calidad de caminería, habilitada o no, incidentes reportados, etc.) de modo de poder generar una capa de inteligencia extra sobre lo que ofrecen las alternativas que manejan actualmente, mostrando cómo con la integración de tecnologías y la optimización adecuada de los datos, es posible generar servicios de valor para la empresa.

3 Requerimientos

3.1 Vista de Casos de Uso

Para esta prueba de concepto se buscan satisfacer la siguiente lista de requerimientos básicos:

1) Dado un punto de geolocalización para la retirada de alimentos (pick-up), y uno o varios de entrega (drop-offs):

- saber si la ruta ya se hizo, y en caso negativo, generarla
- retornar una lista con las posibles rutas.

2) Persistir las rutas asignándoles un “peso/prioridad” (importancia/valor para el negocio).

3) Definidos los atributos relevantes, alterar en función de los mismos el “peso/prioridad” y registrar los incidentes por los cuales ha aumentado el peso.

4) En caso de registrarse incidentes que hayan aumentado el peso de la ruta, mostrar una descripción del mismo.

5) Mostrar en un mapa OSM la alternativa elegida como ruta entre los puntos de entrega, con la descripción de la misma.

Para esta etapa se tomarán como buenas las secuencias armadas para la distribución, aunque se recomienda fuertemente analizar dichos procesos, que abarcan más que los procesos de ruteo aquí abordados.

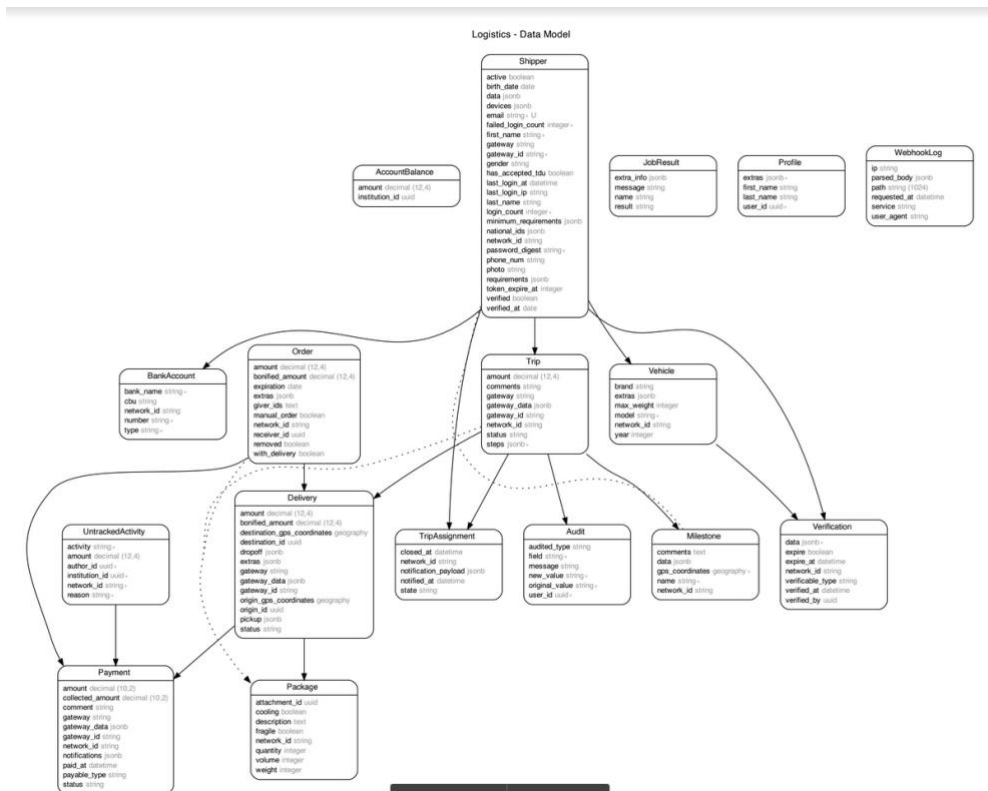
3.2 Vista de Datos

Descripción general

La Empresa maneja diferentes fuentes de datos, a saber:

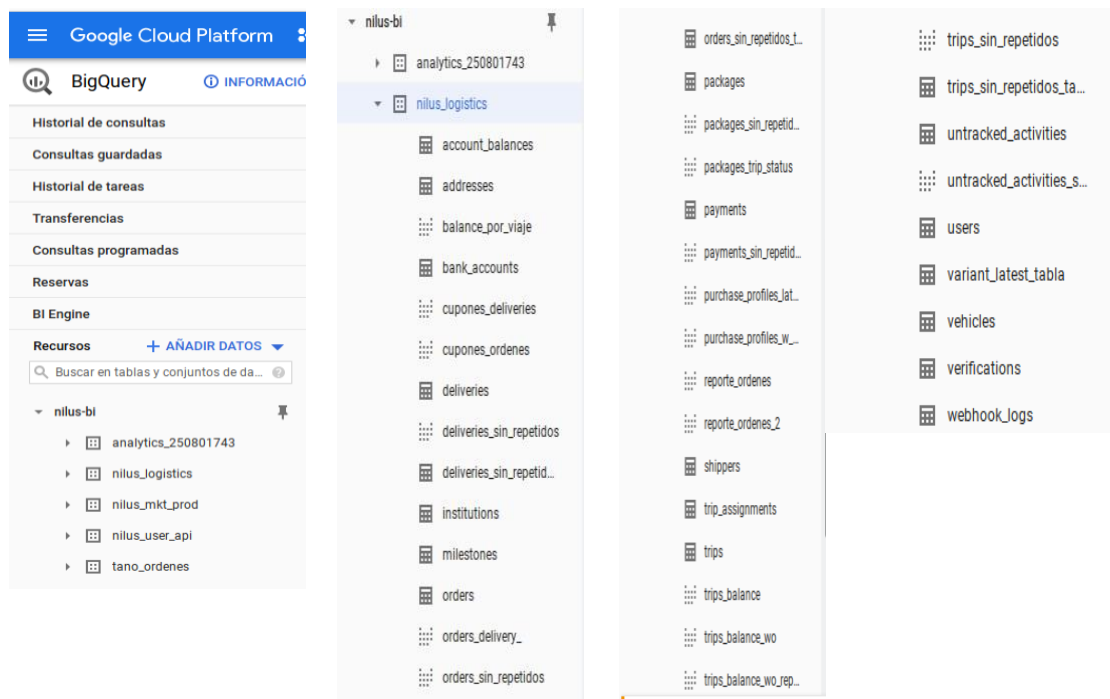
1. Calles y geolocalización: Google Maps, OSM.
2. Direcciones de entrega: hay al menos una por institución, aproximadamente 1000 puntos.

3. Rutas de entrega: hay al menos 1 por orden, hasta 5 generalmente. Hoy día, al tener un centro de distribución centralizado, el destino, productos de la orden, chofer asignado, día y horario de entrega. Histórico de 2 años tomando en cuenta diferentes ciudades. 1300 muestras.
4. Cada ruta tiene N deliveries (órdenes de entrega). Estas son 10k aprox.



La empresa proporcionó inicialmente el diagrama adjunto como para tener una idea de la estructura general de datos que maneja.

Como pudimos comprobar al realizar el análisis, ese fue el esquema original, el cual como es de esperar en un entorno dinámico y cambiante hoy ya ha sido modificado.



Adicionalmente al diagrama general, la empresa también concedió acceso a los datos vía Google Cloud Platform con la herramienta “Big Query”, para permitir el análisis detallado de los mismos.

Del análisis antedicho se resume que los datos relevantes para el presente proyecto están conformados principalmente por unas pocas estructuras residentes en el conjunto de datos denominado “nilus_logistics”, encontrándose principalmente en formatos “Json”, radicando ahí el mayor trabajo y complejidad de extracción.

A continuación, se muestra el total de tablas que componen el conjunto de datos de nuestro interés para luego detallar las estructuras, campos y datos considerados relevantes:

Las tablas relevantes a considerar en una primera instancia del subconjunto “nilus_logistics” son: **addresses, deliveries, milestones, vehicles, trips.**

3.3 Consideraciones y Decisiones Tomadas

De la exploración de datos en Big Query surgieron unas primeras observaciones, que repasamos a continuación de modo no exhaustivo.

- En primer lugar, si bien se estaba utilizando una RDBMS, los datos impactados allí para nuestro caso no estaban normalizados. Los mismos estaban en json, y no teníamos garantías de que estuviesen adecuadamente curados.
- En segundo lugar, observamos que había tablas que eran consultas -similares a las originales- pero en las que supuestamente no había repetidos.
- A simple vista se podía observar que no había uniformidad en el ingreso de los datos, ya a niveles puntuales (ej.: nombres de países), pero también en duplicaciones de registros e incluso a ingresos con distinto id.

Por otro lado, los datos que teníamos obedecían al viejo modelo de negocios, en donde para una orden se recorrían varios puntos de *pickup*. De hecho, en muchos casos observamos que la cantidad de puntos de *pickup* superaba a la de los puntos de entrega o *dropoff*. Por lo tanto, la información de los viajes que se encontraba en la tabla trips tampoco nos iba a ser de utilidad en este caso.

Una vez analizados y depurados los datos, decidimos que lo más sensato era trabajar con un subconjunto de datos muy precisos, al cual le hicimos un proceso de data cleansing puntual y lo guardamos en una serie de archivos json para poder realizar una segunda exploración de datos más certera.

Allí pudimos confirmar que los datos de los viajes -tabla Trips- no nos iban a servir por el cambio de modelo de negocios, y que otros datos -como el de los vehículos- tampoco eran datos confiables, pues al depurarlos vimos que eran realmente escasos. Llegados a este punto estaba claro que lo primero que debería desarrollar la empresa es un proyecto de data cleansing junto con una política de Gobernanza de Datos.

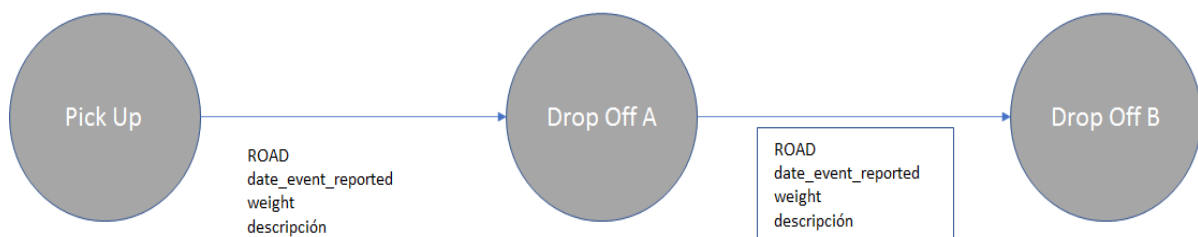
Finalmente, y teniendo en cuenta que se trataba de una prueba de concepto, filtramos los datos de los puntos de entrega de la ciudad de Rosario, quedándonos solamente con la siguiente información de los mismos:

- id de punto de entrega, para poder identificarlo unívocamente (aunque encontramos casos de asignación de id distintos para un mismo centro)
- nombre del centro

- latitud y longitud de ubicación del centro, que estaban registradas juntas, pero deberemos separarlas para poder interaccionar con Open Street Maps
- dirección del centro
- Ciudad, la cual dejamos solo para detectar algún posible mal ingreso a simple vista.

Tomando en cuenta esta estructura para la prueba de concepto decidimos llevar la representación en neo4j a una estructura básica mínima que pueda expandirse.

Estructura Básica de Ruta de Entrega



Desde el punto de vista de la estructura de datos necesaria para el armado de la ruta está claro que no hay diferencias entre los Nodos de Dropoff y de Pick Up (de hecho, éste será uno solo para cada ciudad o región).

Por otro lado, definimos una relación ROAD que identifica cuando existe una conexión o ruta conocida entre dos Nodos. A la misma le agregamos Propiedades básicas para juntar información sobre las mismas, uno de los requerimientos solicitados. Para esta prueba de concepto hicimos una estructura muy básica, la cual consiste en que la relación es una lista que contendrá eventos reportados con esta información:

- `date_event_reported`: registra la fecha del evento sobre la ruta en cuestión
- `weight`: es un valor calculado a definir que asigna un peso a la ruta, y será utilizado por los algoritmos como aStar
- `event_description`: un texto con una breve descripción sobre lo que sucede (ej: corte de calles por obras en tal calle, corte por piquetes, intervenciones policiales, calle X anegada, etc.)

Para la prueba de concepto realizamos una serie de ejemplos pequeños para mostrar cómo funciona. En caso de crecer obviamente será necesaria una tipología adecuada de eventos que contemple la peculiaridad de los casos que se encuentran en las zonas de distribución. Un registro adecuado de las mismas puede ser importante para realizar trabajos analíticos a futuro.

Otro punto a dejar sentado. En el código la función de resolver ruta por OpenStreetMaps (`network_layer.solve`) fue dejado con el parámetro `return_route` en `True`, si bien no hacemos un uso de esos datos. La razón es simple: podemos realizar en neo4j nodos intermedios geolocalizados para cada sub tramo (en el código está comentada la línea `result['routes']['features'][0]['geometry']['paths']`). Como ejemplo, un tramo sencillo de 2 km tenía aproximadamente 40 pares de latitud - longitud que podían guardarse. Se desechó dicha posibilidad pues, más adelante veremos que habría mejores modos de utilizar los servicios de OSM.

En cuanto a los eventos como propiedades de las rutas y el valor de los mismos para el proceso de distribución en sí, debemos destacar dos puntos distintos pero complementarios:

- Dada la naturaleza de los lugares de distribución, es importante poder registrar eventos peculiares de cada zona, probablemente difícilmente capturables en estructuras generales. Por ello pensamos en una estructura pequeña y abierta, customizable para cada caso. Ejemplo, no es el mismo tipo de incidentes el que se puede encontrar en la periferia de Montevideo al que se puede encontrar en alguna ciudad de Chiapas o en Nigeria (lugares donde la empresa piensa expandirse)
- Por otro lado, sí es importante tener en cuenta estructuras más estandarizadas de metadata para estos casos. Para ello OSM cuenta con un marco de referencia importante [3] que implementa para sus métodos para flotas. Solo a modo de ejemplo, en el parámetro `Restrictions`, la tabla incluye valores como “Avoid Unpaved Roads”, “Driving a Truck”, “Use Preferred Truck Routes”, etc.

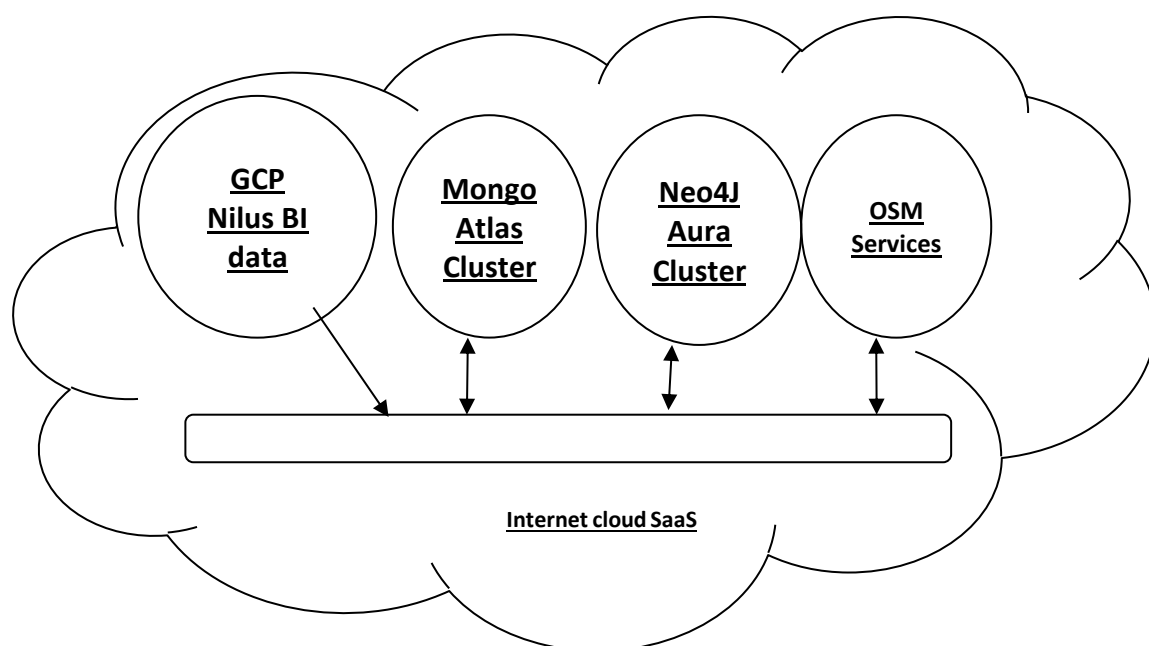
Es importante tener en cuenta esta división de trabajo entre lo peculiar, dado por la naturaleza de las zonas de distribución y sus incidentes, y por otro lado aquellos servicios estandarizados de los que se puede consumir. El primer caso agregará una capa de información contextual de suma utilidad para el caso puntual; en el segundo caso es imprescindible suscribirse a estándares en uso y utilizar los servicios ya existentes.

En el Anexo sobre importación y transformación de los datos a MongoDB se verán claros algunos ejemplos de lo hablado, lo mismo que un detalle más exhaustivo en el Anexo sobre la exploración en Big Query.

3.4 Cronograma

Trimestre 1	2020										
Octubre				Noviembre				Diciembre			
Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4
Aceptación de Anteproyecto	Validación de Arquitectura	Validación de Arquitectura	Validación de Arquitectura	Validación de Arquitectura	Validación de Arquitectura	Detalle Arquitectura	Detalle Arquitectura	Detalle Arquitectura	Detalle Arquitectura		
Visión general de la Solución	Relevamiento de Datos	Relevamiento de Datos	Relevamiento de Datos	Relevamiento de Datos	Relevamiento de Datos	Análisis Detallado Datos	Análisis Detallado Datos	Análisis Detallado Datos	Análisis Detallado Datos		
Formalización de Proyecto	Análisis inicial de datos	Análisis inicial de datos	Análisis inicial de datos	Análisis inicial de datos	Análisis inicial de datos	Selección Tecnologías	Selección Tecnologías	Selección Tecnologías	Selección Tecnologías		
Evaluación de Alcance	Propuestas de Arquitectura	Propuestas de Arquitectura	Propuestas de Arquitectura	Propuestas de Arquitectura	Propuestas de Arquitectura	Pruebas Tecnologías	Pruebas Tecnologías	Pruebas Tecnologías	Pruebas Tecnologías		
Aceptación de Cronograma	Análisis de Tecnologías	Análisis de Tecnologías	Análisis de Tecnologías	Análisis de Tecnologías	Análisis de Tecnologías	Diseño Middleware	Diseño Middleware	Diseño Middleware	Diseño Middleware		
Documentación	Documentación	Documentación	Documentación	Documentación	Documentación	Documentación	Documentación	Documentación	Documentación		
Trimestre 2	2021										
Enero				Febrero				Marzo			
Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4
		Despliegue Arquitectura	Despliegue Arquitectura	Testing Arquitectura	Testing Arquitectura	Testing Arquitectura	Testing Arquitectura	Testing Arquitectura	Testing Arquitectura		
		Pruebas Datos Seleccionados	Pruebas Datos Seleccionados	Implementación Solución Diseñada	Implementación Solución Diseñada	Implementación Solución Diseñada	Implementación Solución Diseñada				
		Pruebas individuales Tecnologías	Pruebas individuales Tecnologías	Ajustes Tecnologías (Mongo - Neo4j, OSM)	Ajustes Tecnologías (Mongo - Neo4j, OSM)	Ajustes Tecnologías (Mongo - Neo4j, OSM)	Ajustes Tecnologías (Mongo - Neo4j, OSM)	Ajustes Tecnologías (Mongo - Neo4j, OSM)	Ajustes Tecnologías (Mongo - Neo4j, OSM)		
		Documentación	Documentación	Documentación	Documentación	Documentación	Documentación	Documentación	Documentación	Documentación	Documentación

4 Arquitectura de la solución



La solución estará soportada íntegramente en servicios de TI sobre internet.

La empresa provee el acceso a la fuente de los datos sobre tablas existentes en Google Cloud Platform,[7] desde allí se toman los datos necesarios para que persistan en un cluster montado en el servicio Atlas de Mongo DB.[5] Una vez realizados los trabajos de adecuación de los datos (Normalización, ETL Feature Engineering, etc.), se irán almacenando las rutas en Neo4j [2] montado también en una instancia del servicio de cluster Aura de Neo, previa consulta con los servicios de OpenStreetMap (OSM)[4]. Toda la interacción con los mencionados servicios se hace mediante scripts en python brindando la flexibilidad de instalar o “dejar corriendo”, cada parte del “middleware” en acuerdo con el cliente y la solución particular.

Finalmente se utilizan nuevamente los servicios de OSM para mostrar en un mapa del Web Browser la ruta destacada en el mapa y obtener las indicaciones específicas.

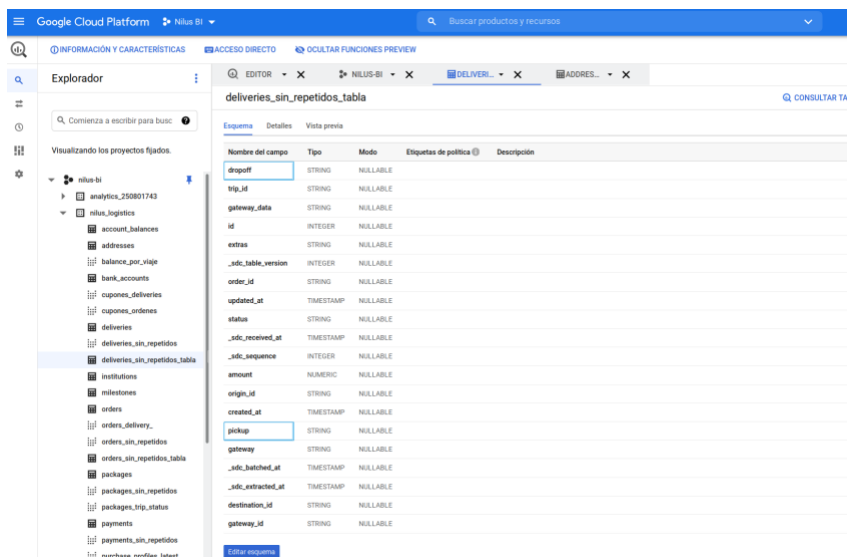
Dado el estado de curado de los datos de la empresa, hay una parte de la arquitectura que no se encuentra on line, pero que en un proceso en producción tendría que estarlo. Nos referimos al proceso de importación y curado de datos de Big Query. En este caso puntual hemos exportado a archivos json para un proceso de data cleansing final y ser impactados en MongoDB[5].

5 Tecnologías seleccionadas

En la implementación de la prueba de concepto se han utilizado las tecnologías descritas en el apartado anterior. Aquí procederemos a dar un detalle sobre la implementación específica de dichas tecnologías en el caso concreto.

5.1 Descripción general

Resulta importante mencionar cómo concepto general en lo que respecta a la selección de las tecnologías, que un aspecto clave lo conforma la disponibilidad actual de un ecosistema de productos y soluciones que permite una gran flexibilidad a la hora de elegir las mismas para su implementación. Lo antedicho hace que, de acuerdo a la realidad relevada en referencia a la empresa, su actualidad, posibilidades, lineamientos y planes de desarrollo sea posible adaptarse a las mismas con gran dinamismo.



The screenshot displays the Google Cloud Platform interface for BigQuery. On the left, the 'Explorador' (Explorer) pane shows a project named 'nibus-bi' with a folder 'analytics_350801743' containing a table 'deliveries_sin_repetidos.tabla'. The main area shows the 'Esquema' (Schema) tab for this table. The schema is a table with columns: 'dropoff', 'trip_id', 'gateway_data', 'id', 'extras', '_sdc_table_version', 'order_id', 'updated_at', 'status', '_sdc_received_at', '_sdc_sequence', 'amount', 'origin_id', 'created_at', 'pickup', 'gateway', '_sdc_batched_at', '_sdc_extracted_at', 'destination_id', and 'gateway_id'. Each column has a 'Tipo' (Type) and a 'Modo' (Mode) of 'NULLABLE'.

Nombre del campo	Tipo	Modo	Etiquetas de política	Descripción
dropoff	STRING	NULLABLE		
trip_id	STRING	NULLABLE		
gateway_data	STRING	NULLABLE		
id	INTEGER	NULLABLE		
extras	STRING	NULLABLE		
_sdc_table_version	INTEGER	NULLABLE		
order_id	STRING	NULLABLE		
updated_at	TIMESTAMP	NULLABLE		
status	STRING	NULLABLE		
_sdc_received_at	TIMESTAMP	NULLABLE		
_sdc_sequence	INTEGER	NULLABLE		
amount	NUMERIC	NULLABLE		
origin_id	STRING	NULLABLE		
created_at	TIMESTAMP	NULLABLE		
pickup	STRING	NULLABLE		
gateway	STRING	NULLABLE		
_sdc_batched_at	TIMESTAMP	NULLABLE		
_sdc_extracted_at	TIMESTAMP	NULLABLE		
destination_id	STRING	NULLABLE		
gateway_id	STRING	NULLABLE		

En nuestro caso particular la empresa como ya hemos mencionado tiene una alianza estratégica y trabaja con las herramientas de la plataforma de Google (Google Cloud Platform). Específicamente para nuestro caso de uso, utilizaremos la plataforma “BigQuery” dentro de GCP para poder realizar la exploración de los datos a los que nos han dado acceso.

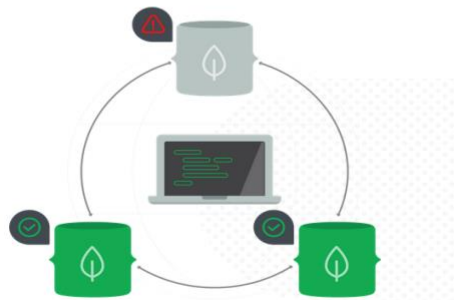
MongoDB Atlas

Cloud-hosted MongoDB service on AWS, Azure and Google Cloud. Deploy, operate, and scale a MongoDB database in just a few clicks



Always on and durable

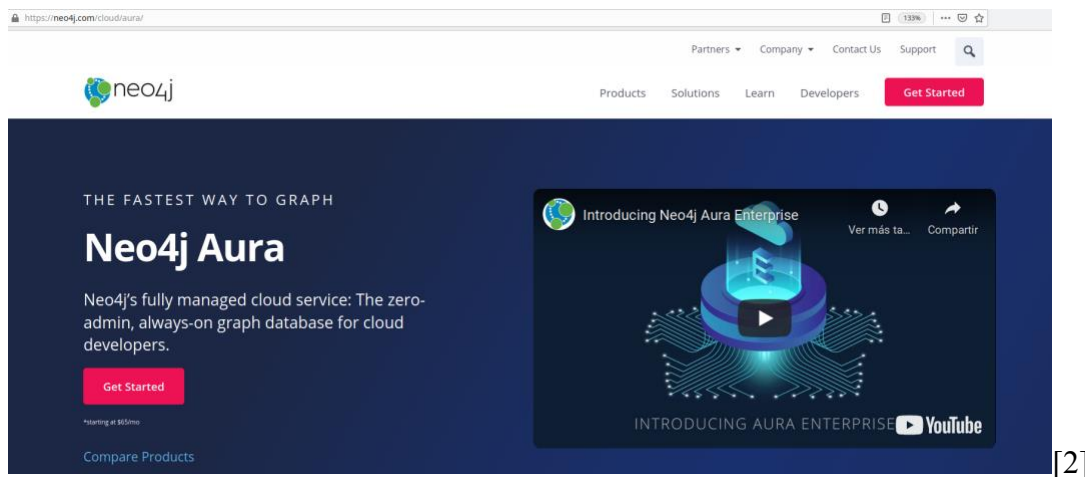
With MongoDB Atlas, your self-healing clusters are made up of geographically distributed database instances to ensure no single point of failure. Want even better availability guarantees? Get multi-region fault tolerance by enabling cross-region replication. MongoDB Atlas also includes powerful features to enhance reliability for your mission-critical production databases, such as continuous backups and point-in-time recovery.



[10]

Luego de seleccionados los datos relevantes para nuestro trabajo persistiremos los mismos en una instancia clusterizada de MongoDB en su versión como SaaS (Software as a service), en Mongo Atlas en la nube, fundamentalmente por la flexibilidad que permite la herramienta, la disponibilidad del acceso a la infraestructura ya montada en la nube y la curva de aprendizaje dadas los conocimientos y experiencias de quienes estamos llevando adelante el proyecto.

Para el guardado de la información “extra” de las incidencias en las rutas retornadas por OSM se optó por utilizar la tecnología de una base de datos de grafos como Neo4j principalmente porque es una tecnología que promete mucho sobretodo a la hora de descubrir relaciones “ocultas” entre los datos y que podrían aportar en un futuro, más allá de la implementación puntual de este caso particular, aspectos importantes a la hora de desarrollar soluciones para el negocio que permitan explotar en mayor medida aspectos de aprendizaje automático y predicciones.



[2]

En el momento de la realización del presente trabajo, no se disponía de una capa gratuita (se encuentra “en construcción”), de esta versión de Neo4j (denominada Aura y que es la versión SaaS (Software as a service) de la herramienta en su versión de nube clusterizada), por lo que se contrató para este trabajo la capa inicial mínima que permitiera implementar la prueba de concepto.

Para toda la información de geolocalización, es decir los mapas, como ya se mencionó, se utilizó principalmente por la facilidad y disponibilidad de acceso a la información todo el ecosistema provisto por OpenStreetMap [8]. La empresa también sugirió desde el inicio del presente trabajo la utilización de dicha opción debido a su experiencia con el uso de otras tecnologías como Google Maps y su falta de precisión y/o flexibilidad a la hora de trabajar con zonas periféricas como es su caso.



OpenStreetMap lo crea una gran comunidad de colaboradores que aportan y mantienen datos sobre caminos, senderos, cafeterías, estaciones de ferrocarril y muchas cosas más a lo largo de todo el mundo.

📍 Conocimiento local

OpenStreetMap valora mucho el conocimiento local. Los colaboradores utilizan imágenes aéreas, dispositivos GPS, mapas y otras fuentes de datos libres para verificar que los datos de OSM sean precisos y estén actualizados.

👥 Impulsado por la comunidad

La comunidad de OpenStreetMap es diversa y entusiasta y crece todos los días. Entre nuestros colaboradores figuran cartógrafos apasionados, profesionales de GIS, ingenieros que hacen funcionar los servidores de OSM, humanitarios que elaboran mapas de zonas de desastre, y muchas personas más. Para obtener más información sobre la comunidad, véase el [Blog de OpenStreetMap](#), [diarios de los usuarios](#), [blogs comunitarios](#), y el sitio web de la [Fundación OSM](#).

© Datos abiertos

OpenStreetMap es *datos abiertos*: puede usarlo libremente para cualquier propósito siempre que dé crédito a OpenStreetMap y a sus colaboradores. Si altera o se basa en los datos de alguna manera, sólo puede distribuir el resultado bajo la misma licencia. Consulte la [página sobre Derechos de autor y Licencia](#) para obtener más detalles.

💰 Legal

Este sitio y muchos otros servicios relacionados son gestionados formalmente por la [Fundación OpenStreetMap](#) (OSMF) en nombre de la comunidad. El uso de todos los servicios gestionados por la OSMF está sujeto a nuestros [Términos de uso](#), [normativa de uso aceptable](#) y nuestra [normativa de privacidad](#).

[Comuníquese con la OSMF](#) si tiene preguntas sobre licencias, derechos de autor u otras cuestiones legales.

OpenStreetMap, el logotipo de la lupa y State of the Map son [marcas comerciales registradas de OSMF](#).

👤 Socios

El alojamiento es apoyado por [UCL](#), [Bytemark Hosting](#), y otros [socios](#).

[4]

5.2 Base de Datos Documental

BigQuery [7] es la herramienta de GCP utilizada por la empresa para realizar las consultas SQL sobre los datos. La exploración de las diferentes estructuras muestra que los datos relevantes para el trabajo se encuentran almacenados en formato json dentro de campos

generales. Esto llevó, como primera decisión estratégica, a la utilización de una base de datos documental (MongoDB) como paso intermedio para la solución, en la cual se realizará el proceso de ETL necesario para continuar el proceso.

Luego el almacenamiento de los lugares de despacho -uno solo para el caso- y entrega junto con las mejores rutas se almacena en Neo4j que por tratarse de una base de datos orientada a grafos nos provee la posibilidad de dar semántica a las relaciones entre nodos, distinguiendo tipos y propiedades de las relaciones, así como el acceso optimizado a los algoritmos para las búsquedas de los caminos entre los nodos (path finding algorithms, detallados en los anexos y en la bibliografía).

Se utilizarán los servicios de OSM para la obtención de los datos de las rutas con sus coordenadas válidas de geolocalización para cada momento que la ruta específica -con su cantidad de paradas- sea solicitada.

Por último, no implementado en la prueba de concepto y sí sugerido para una ulterior fase, se recomienda también la utilización de los servicios de optimización de rutas en las capas pagas de OSM, de modo de poder evaluar periódicamente la eficiencia de las rutas almacenadas en neo4j, como modo de optimizar el servicio de distribución.

5.3 Middleware

Hemos utilizado Python 3 como lenguaje de scripting para los distintos casos descritos. Como estamos en una prueba de concepto hemos utilizado la distribución de Anaconda sobre Jupyter Notebooks para correr y analizarlos independientemente.

En los Anexos se encuentran los exports de los Notebooks realizados, con los comentarios específicos para cada caso. En total son tres scripts:

- un script que importa los datos traídos de BigQuery, los transforma hacia la estructura requerida para la prueba y los persiste en MongoDB
- un script que exporta los datos desde MongoDB hacia Neo4j. En el mismo hacemos un juego de datos de pruebas de rutas.
- un script que recupera las rutas de neo4j, les pasa las coordenadas a OpenStreetMap para el trazado de la misma

5.4 Interfaces

Las interfaces utilizadas en cada caso fueron analizadas de acuerdo con la documentación específica en cada caso, e implementadas aquellas más recomendadas para python. En la bibliografía está detallada la documentación para cada caso.

- **BiqQuery**

Para el caso de BigQuery hemos hecho un export directo a csv, por lo cual no nos hemos conectado a ninguna interfaz específica del servicio desde python. Este, obviamente, es uno de los temas a evaluar si hay una etapa posterior de trabajo.

- **MongoDB**

Para la conexión de python con MongoDB utilizamos de la distribución PyMongo la biblioteca pymongo, la cual nos da el driver nativo para MongoDB.

- **Neo4j**

Para la conexión a Neo4j desde Python utilizamos la biblioteca GraphDatabase, la cual tiene soporte específico de neo4j.

Por otro lado, utilizamos la biblioteca apoc, la cual tiene -entre otros- procedimientos de enrutamiento que también se conectan a OpenStreetMaps, puntualmente hablamos de la implementación algoritmos como allSimplePaths, aStar o dijkstra.

- **OpenStreetMap**

Para la conexión con OpenStreetMap nos decantamos por ArcGIS para python, la cual tiene abundante documentación y soporte, lo mismo que ofrece gran cantidad de servicios en distintas capas, lo cual permite escalar en la solución si así es deseado.

Puntualmente hemos utilizado los servicios abiertos de la capa network.RouteLayer, pero recomendando el futuro análisis y utilización de los servicios de network.analysis, los cuales son de una capa especializada -entre otros puntos- en los problemas de ruteo y manejo de flotas.

6 Conclusiones

El presente trabajo llevó adelante la implementación de una prueba de concepto de tecnología de la información en el contexto del trabajo final de la maestría en la especialización de BigData con el objetivo de mostrar las posibilidades que puede brindar el uso de estas tecnologías y estrategias en pos de la resolución y apoyo para el abordaje de una problemática determinada en una empresa real.

Finalizadas las diferentes etapas llevadas a cabo hemos podido confirmar una vez más algo que por la experiencia de los integrantes ya se tenía meridianamente claro, y consiste en que la calidad de cualquier implementación en tecnologías de la información está directamente relacionada con la calidad de los datos que se manejan en el ámbito de aplicación. Por lo antedicho de querer implementar las soluciones mostradas como posibles en el presente trabajo, el paso inicial de cualquier trabajo futuro será comenzar por realizar un trabajo profundo en todo lo que refiere a la recolección, mantenimiento y limpieza de los datos, en pos de su calidad y veracidad, desde las interfaces y proceso de recolección de los mismos como la adecuación de las estructuras y tecnologías definidas para su almacenamiento y procesamiento.

Durante el proceso de análisis detallado de los datos del presente trabajo se pudo comprobar una vez más que los temas de calidad e integridad de los datos la gran mayoría de las veces, y esta no es una excepción, son relegados en pos de la funcionalidad de la implementación de los sistemas que soportan el negocio. Lo anterior hace que, si bien se puede tener un gran dinamismo a la hora de salir al mercado con una solución y/o estrategia de negocios, luego al intentar crecer sobre la misma y al no estar lo suficientemente sólidos los cimientos basados en los datos, sea muy difícil o casi imposible proyectar crecimiento sostenido sobre los mismos.

Otra conclusión importante de la que se quiere dejar constancia y que ha dejado como enseñanza o reafirmado la implementación del presente trabajo, es que para que cualquier solución tecnológica tenga éxito, y en el caso de soluciones de tecnologías de la información basadas en BigData y aprendizaje automático es un aspecto, que más que importante se torna fundamental, es que la implementación de la misma debe estar enmarcada en un proceso general de incorporación, aceptación y cambio cultural llevado adelante en forma transversal en todas las áreas y sectores que conforman la organización, liderada desde y por los ámbitos gerenciales.

Se deja constancia de que es posible mejorar la metadata que se le puede mandar a OSM que cuenta con servicios disponibles de valor y costo agregado o haciendo el proceso “on premises” u “on cloud”, en una arquitectura apropiada con costos mayores pero que escapan al alcance del presente trabajo.

7 Recomendaciones

Como recomendaciones finales elaboramos la siguiente lista que a juicio de los autores, si se tiene en cuenta, podría aportar en oportunidades de mejora tanto si la empresa decidiera implementar en la práctica algún tipo de solución basada en el presente trabajo o bien para evaluar en alguno de sus procesos actuales basados en tecnologías de la Información:

- **Calidad, integridad y accesibilidad de los Datos**

Como ya se mencionó la calidad de la materia prima que hará posible crear el negocio, es decir los datos, debe estar garantizada desde su recolección tanto sea en la implementación de controles apropiados en las interfaces de recogida de los mismos en las aplicaciones propietarias como también en las obtenidas a través de servicios de terceros.

- **Selección de Tecnologías**

Las tecnologías que sean seleccionadas no deben ser evaluadas con sesgo exclusivo por la riqueza o calidad de sus funcionalidades sino también de su universalidad de acceso, coste, ciclo de actualización, proyección a futuro, curva de aprendizaje, disponibilidad del personal capacitado y comunidad de soporte. Tal vez lo mejor sea elaborar una paramétrica que aúne todos o algunos de estos aspectos y otros que se entiendan como relevantes por los responsables.

- **Flexibilidad en implementación**

En los últimos 30 años hemos observado el pasaje desde soluciones propietarias a un ecosistema de soluciones multi vendedor y abiertas propiedades de “la comunidad”. Lo anterior si se sabe explotar redundará el “no atarse” a soluciones monolíticas que no pueden acompañar la dinámica de los mercados. Cualquier solución del siglo XXI debería tenerse en cuenta y explotar lo antedicho en pos de la flexibilidad de implementación.

- **“Time to Market” con visión de mediano y largo plazo**

La velocidad de implementación no debería hacerse hipotecando las chances de crecimiento y éxito en el mediano y largo plazo. Dicho de un modo sencillo y aceptado, lo urgente no debe postergar lo importante.

- **Mejora continua como estrategia transversal**

El proceso de incorporación, mantenimiento y madurez en el uso de las tecnologías que soportan la estrategia de negocio debe estar enmarcado y “ser parte de” un proceso de mejora continua de toda la organización. [11]



8 Lecciones aprendidas

- **Entorno empresarial**

La naturaleza dinámica del entorno empresarial y competitivo hace que por lo general las estrategias de negocio y su tiempo de salida al mercado presionen para que los equipos de TI provean soluciones y herramientas de calidad en el menor tiempo posible. Si bien lo anterior no es una novedad, la realidad actual en lo que refiere a la gran disponibilidad de herramientas y tecnologías hace muy fácil integrar las mismas y proveer soluciones de forma muy rápida. Lamentablemente si no se siguen ciertas estrategias básicas a la hora de la selección e implementación de las arquitecturas tecnológicas las soluciones rara vez puedan escalar y acompañar el crecimiento de los negocios en forma ordenada y/o por lo menos controlada. Por lo antedicho un aprendizaje reafirmado en el presente proyecto es tener siempre claro que no importa lo apremiante que sean las circunstancias a la hora de armar el “stack” tecnológico, en algún momento si este no ha sido desarrollado pensando en el crecimiento y mantenimiento futuro, en algún momento y antes de que el mismo se convierta en un impedimento deben tomarse las acciones correctivas necesarias.

- **Ecosistema tecnológico**

La gran disponibilidad de herramientas y metodologías que existen actualmente hace que muchas veces no se tenga en cuenta las grandes diferencias en lo que respecta a la calidad, la disponibilidad y el soporte existente para las mismas. La existencia de entornos donde no existe costo inicial de ingreso a determinados servicios y tecnologías provoca que muchas decisiones con respecto a si elegir tal o cual herramienta y/o proveedor se realizan teniendo únicamente en mente la funcionalidad y dichos costos iniciales y no los costos “ocultos” generalmente de adopción, mantenimiento, curva de aprendizaje y disponibilidad de personal calificado. Por todo lo antedicho, se debe tener presente que no existen en ningún caso “soluciones mágicas”, “baratas” y/o que cumplen el 100% de las maravillas que prometen. A la hora de seleccionar el Stack tecnológico se deben tener en cuenta entonces los aspectos mencionados teniendo siempre en cuenta que una solución para una empresa en un momento determinado puede no servir para otra muy similar pero que tiene un entorno diferente, ya sea, contexto geográfico, a su evolución en la adopción de las tecnologías, etc.

- **Planes, Ideas y realidad**

Al comienzo del presente trabajo se comenzó con una idea y determinados planes para su ejecución. Al no tratarse de un trabajo puramente académico donde generalmente se parte de supuestos que pueden ser respetados hasta el final del proyecto, sino basados en la realidad de una empresa existente, el equipo de trabajo, al profundizar el análisis se fue encontrando con una realidad que lejos de ser la ideal obligó a transformar el objetivo inicial hasta reducirlo para poder cumplir con los plazos establecidos. La experiencia indica que siempre que se trate de un caso real, el relevamiento inicial si se hace en forma correcta devela una realidad que muchas veces no es percibida por quienes la viven a diario como tan alejada de lo “ideal” lo que hace cambiar los planes e ideas iniciales. Por lo antedicho siempre es preferible ir “de menos a más”, comenzando con pequeñas pero importantes metas y no trazar de entrada objetivos que sean tan ambiciosos que tornen cada paso tan pequeño dando la sensación de que el objetivo real está muy lejos de la meta a conseguir.

- **Los grupos de trabajo y el factor humano**

Un aspecto no menor y que en este trabajo quedó manifestado una vez más, por ello no queríamos dejar de mencionar, lo constituye la dinámica del grupo de trabajo creado para la consecución de un objetivo y su factor humano. En el caso de estudio lo constituyó la gente de la empresa y nuestro grupo. Como ya hemos mencionado el trabajo fue posible gracias a que la empresa brindó y mostró gran interés en todo momento en apoyar el trabajo dando acceso a sus técnicos y a los datos. El aspecto clave que queremos resaltar consiste en que se debe tener en cuenta que sobretodo a la hora de presentar resultados, y definir los pasos a futuro, las recomendaciones particulares sobre el cómo y en qué momento, deben ser una construcción en conjunto de forma que todos los actores sientan que “son parte de” y no que “el cliente” sienta que “gente de afuera” le viene a decir que y como se deben hacer las cosas. Esto último es fundamental si se pretenden construir soluciones que realmente aporten y puedan llegar a sentirse como propias por parte de quienes van a convivir con ellas a diario.

9 Referencias bibliográficas

- [1] Nilus Global LLC, “Nilus”, nilus.online, 2021 [En línea]. Disponible en: <https://www.nilus.online/> (accedido oct. 24, 2021).
- [2] Neo4j, Inc. “Neo4j Documentation, Neo4j Graph Database Platform.”, 2021 [En línea]. Disponible en: <https://neo4j.com/docs/> (accedido oct. 24, 2021).
- [3] Esri Global. “arcgis.network module — arcgis 1.9.1 documentation”, 2021 [En línea]. Disponible en: <https://developers.arcgis.com/python/api-reference/arcgis.network.toc.html> (accedido oct. 24, 2021).
- [4] Community-owned, supported by OpenStreetMap Foundation. “OpenStreetMap”, 2021 [En línea]. Disponible en: OpenStreetMap. <https://www.openstreetmap.org/> (accedido oct. 30, 2021).
- [5] MongoDB Inc. “MongoDB Documentation”. 2021[En línea]. Disponible en: <https://docs.mongodb.com/> (accedido oct. 24, 2021).
- [6] Python Software Foundation. “Python.org”, 2021[En línea]. Disponible en: <https://www.python.org/shell/> (accedido nov. 01, 2021).
- [7] Alphabet. “Cloud Computing Services”, Google Cloud 2021 [En línea]. Disponible en: <https://cloud.google.com/> (accedido oct. 30, 2021).
- [8] José Ortiz y Sanz, José Ortiz y Sanz, Instituciones de Arquitectura Civil Acomodadas en lo posible a la doctrina de Vitruvio (1819) - Referenciado en “Los diez libros de Arquitectura” Vitruvio., 10 vols. Madrid, Imprenta Real. 1787. [En línea]. Disponible en: https://www.sedhc.es/biblioteca/tratado.php?ID_pubD=39
- [9] Alphabet. “Google Maps”, Google online Services 2021 [En línea]. Disponible en: <https://cloud.google.com/maps> (accedido oct. 30, 2021).
- [10] MongoDB Inc. “The MongoDB 5.0 Manual — MongoDB Manual”. 2021 [En línea]. Disponible en: <https://docs.mongodb.com/manual/> (accedido oct. 24, 2021).

- [11] “El círculo de Deming o la espiral de mejora continua”, Asesorías. 2021 [En línea]. Disponible en: <https://asesorias.com/empresas/modelos-plantillas/circulo-deming/> (accedido oct. 29, 2021).
- [12] B. Hackett, "pymongo: Python driver for MongoDB" 2021 [En línea]. Disponible en: <http://github.com/mongodb/mongo-python-driver> (accedido: oct. 24, 2021).
- [13] Neo4j, Inc. “The Neo4j Graph Data Science Library Manual v1.7 - Neo4j Graph Data Science” 2021[En línea]. Disponible en: <https://neo4j.com/docs/graph-data-science/1.7/> (accedido oct. 30, 2021).

10 ANEXO I - Persistencia de datos en MongoDB

En este documento dejaremos constancia de los pasos seguidos para la creación, a partir de los datos exportados en formato json de la base de datos operaciones de Nilus en Bigquery, de una base de datos documental en MongoDB.

Como etapa inicial estableceremos que cada entidad reconocida para nuestro caso será una Colección de Documentos en MongoDB. En esta fase se hace un proceso básico de curado y limpiado de datos con el fin que la información que posteriormente vaya hacia Neo4j sea confiable.

Trabajaremos desde python con las librerías json y pymongo,[12] las cuales nos permitirán trabajar con los datos exportados en json para luego, mediante pymongo, interactuar con la base MongoDB.

```
#conda install -c anaconda dnspython
```

```
#import libraries
```

```
import json
from pymongo import MongoClient
import dns
```

• Creación de Base de Datos Documental en MongoDB

Para la exploración de datos primero desarrollaremos una base de datos local desde donde importar los datos que fueran extraídos en formato json de la base de datos operativa de Nilus desarrollada en BigQuery.

La base creada se llamará nilusdb, y tendrá colecciones de los distintos objetos reconocibles en los datos proporcionados, a saber: viajes (trips), entregas (deliveries), transporte (vehicules) y seguimiento (milestones).

```
[4]: #check that de database was created

print(nilusClient.list_database_names())

['Nilust_Test', 'sample_airbnb', 'sample_analytics', 'sample_geospatial',
'sample_mflix', 'sample_restaurants', 'sample_supplies', 'sample_training',
'sample_weatherdata', 'admin', 'local']

[5]: # Create collections

Collection = db["deliveries"]
Collection_vehicles = db["vehicles"]
Collection_trips = db["trips"]
Collection_milestones = db["trips"]

[6]: filter = {}
Collection.delete_many(filter)

[6]: <pymongo.results.DeleteResult at 0x15439a06848>
```

• Carga y Exploración de Datos

Procederemos a cargar los datos obtenidos de BigQuery en las colecciones ya generadas en nilusdb, y haremos una pequeña exploración de los mismo a fin de tener un panorama algo más claro del tipo de situaciones que nos vamos a encontrar.

• Deliveries

Comenzaremos por cargar los datos de uno de las candidaturas a centro del modelo a desarrollar, los deliveries o entregas.

```
In [7]: 1 # cargamos los datos
2 with open('NilusDeliveries.json', encoding='utf-8') as file:
3     file_data = json.load(file)
4
5 # insertamos los documentos
6 if isinstance(file_data, list):
7     print("many lines")
8     Collection.insert_many(file_data)
9 else:
10    print("only one line")
11    Collection.insert_one(file_data)
12

many lines
```

Evaluaremos algunos de los principales datos de la colección, a saber, cuántos documentos tenemos, cuántos puntos de delivery distintos y cuántos países hay en los datos obtenidos.

```
In [8]: 1 #check the number of documents of the collection
2
3 n_docs = db.deliveries.count_documents({})
4 print(n_docs)
5
6 #check the number of different delivery id
7 print(len(db.deliveries.distinct("id")))
8
9 #check the number of countries in the collection
10 db.deliveries.distinct("dropoff.address.country")

12261
12261

Out[8]: [None,
        'AR',
        'Ar',
        'Argentina',
        'BO',
        'BR',
        'CL',
        'FR',
        'MEX',
        'MEXICO',
        'MX',
        'Mexico',
        'México',
        'PR',
        'Puerto rico',
        'Puerto Rico',
        'US',
        'VE',
        'arg',
        'puerto rico']

In [9]: 1 #número de direcciones de entrega
2 len(db.deliveries.distinct("dropoff.address.id"))

Out[9]: 2034
```

```
In [10]: 1 #número de direcciones de pick up
2 len(db.deliveries.distinct("pickup.address.id"))

Out[10]: 130
```

Tomando en cuenta características generales, se observa que tenemos la información de 24522 deliveries, los cuales se encuentran en 9 países, pero se observa claramente que no hay un criterio unificado en el ingreso de datos (México es nombrado de 5 modos distintos, Argentina de 4), y a su vez no se sigue un criterio estándar de nomenclatura, ingresándose o bien por un sistema de dos caracteres (AR, MX, etc.) o bien por alguna forma no estandarizada del nombre en español.

Por otro lado, cuando tomamos al azar una ciudad, vemos que existe un fuerte desbalance entre los pick up y los dropoff. Está claro que no tienen porqué ser simétricos (y menos ahora que la empresa va hacia un sistema unificado de pick up), pero los desbalances en algunos casos son claros.

Para SAN FERNANDO, por ejemplo, se dan 193 pick ups y tan solo 4 dropoffs. Para Rosario, cuya cantidad de deliveries es muy superior, los números se vuelven de un orden más normal, 3921 pickups frente a 1987 dropoffs.

De modo general vemos que en toda la colección tenemos 130 direcciones de pickup frente a 2034 de dropoff, por lo cual resulta claro que este tipo de desbalances tendrá que ser observado de cerca por si indica algún problema en la calidad de los datos.

Observamos algunos datos más sobre los puntos de entrega.

```
In [45]: 1 #check some data of nested documents in order to see if everything is working
         2
         3 db.deliveries.count_documents({"dropoff.address.city":"San Fernando"})
         4
Out[45]: 4

In [12]: 1 db.deliveries.count_documents({"dropoff.address.city":"Rosario"})
Out[12]: 1987

In [13]: 1 db.deliveries.count_documents({"pickup.address.city":"San Fernando"})
         2
Out[13]: 193

In [14]: 1 db.deliveries.count_documents({"pickup.address.city":"Rosario"})
Out[14]: 3921

In [15]: 1 len(db.deliveries.distinct("id"))
Out[15]: 12261
```

• Transporte

No hemos encontrado mucha información acerca de los vehículos como tampoco de sus características.

En esta fase de prueba de concepto, no parece pertinente incluir en el modelo hasta tener claro cómo va a decantar este aspecto.

```
In [17]: 1 # cargamos los datos
         2 with open('MilusVehicles.json',encoding='utf-8') as file:
         3     file_data_vehicle = json.load(file)
         4 # insertamos los documentos
         5 if isinstance(file_data, list):
         6     print("many lines")
         7     Collection_vehicles.insert_many(file_data_vehicle)
         8 else:
         9     print("only one line")
        10     Collection_vehicles.insert_one(file_data_vehicle)
many lines

In [18]: 1 db.vehicles.count_documents(filter)
Out[18]: 78

In [20]: 1 db.vehicles.count_documents({"extras.features":"refrigerated"})
Out[20]: 24
```

• Viajes (Trips)

Analizaremos los viajes, más que nada para comprender el negocio. Teniendo en cuenta el cambio de modelo operado, con la centralización de centros de distribución por ciudad, muchos de estos viajes ya no tienen sentido como base de información, ya que -como vimos- existían más cantidad de puntos de pick up que de dropoff por viaje.

```
In [44]: 1 # Cargamos los datos
2 with open('NilusTrips.json',encoding='utf-8') as file:
3     file_data_trips = json.load(file)
4 # insertamos los documentos
5 if isinstance(file_data_trips, list):
6     print("many lines")
7     Collection_trips.insert_many(file_data_trips)
8 else:
9     print("only one line")
10    Collection_vehicles.insert_one(file_data_trips)

many lines
```

Observamos la cantidad general de viajes que se han hecho, así como la distribución geográfica. En todos los casos observamos que los datos no están curados y hay distintos criterios para los mismos datos.

```
In [22]: 1
2 db.trips.count_documents(filter)
```

Out[22]: 2616

```
In [23]: 1 # observamos en qué países hay envíos
2 db.trips.distinct("steps.address.country")
```

Out[23]: [None,
'AR',
'Argentina',
'BO',
'BR',
'CL',
'FR',
'MEXICO',
'MX',
'Mexico',
'México',
'PR',
'US',
'arg']

```
In [24]: 1 # observamos en qué estados hay envíos
2 db.trips.distinct("steps.address.state")
```

Out[24]: ['Bayamón',
'Bs As',
'Buenos Aires',
'Buenos aires',
'CIUDAD DE MEXICO',
'Calle Guadalupe',
'Chubut',
'Ciudad de Mexico',
'Ciudad de México',
'Cuauhtémoc',
'Departamento de Cochabamba',
'Iztapalapa',
'MAR DEL PLATA',
'Manatí',
'Monte Llano',
'Occitanie',
'PR',
'Palmas',
'Ponce',
'Provincia de Buenos Aires',
'Querétaro',
'Quintana Roo',
'Región de Tarapacá',
'Río Grande',
'Río Jueyes',
'San Juan',
'Santa Fe',
'São Paulo',
'Trujillo Alto',
'Yucatán']

```
In [25]: 1 # miramos en qué ciudades hay viajes
        2 db.trips.distinct("steps.address.city")
```

```
Out[25]: [None,
'- ',
'3 de febrero',
'Acassuso',
'Aldo Bonzi',
'Area de Promoción El Triángulo',
'Avellaneda',
'Barracas',
'Batan',
'Batán',
'Bayamón',
'Beccan',
'Bella Vista',
'Benavidez',
'Berazategui',
'Bernal Oeste',
'CDMX',
'Carlos Spegazzini',
'Cataño',
'Cayey',
'Cercado',
'Ciudad Evita',
'Ciudad de Mexico',
'Ciudad de México',
'Comuna 1',
'Comuna 10',
'Comuna 15',
'Comuna 2',
'Comuna 4',
'Comuna 7',
'Comuna 8',
'Comuna 9',
'Cuauhtemoc',
```

Por otro lado, podemos observar que esos 1308 Trips vistos en los datos se corresponden con 4777 deliveries, suponiendo entonces que cada Trip está compuesto de modo estándar por 3 a 4 Deliveries.

```
In [26]: 1 len(db.trips.distinct("steps.delivery_id"))
```

```
Out[26]: 4777
```

```
In [28]: 1 db.deliveries.count_documents({"dropoff.address.country": "FR"})
```

```
Out[28]: 11
```

Como punto final observamos una serie de documentos referidos a Francia, pero se observa que su entrega es en General Pueyrredón en la Provincia de Buenos Aires. Nuevamente, esto sugiere restringir bastante el uso de los datos de prueba.

```
In [34]: 1 dropoffFrance = list(db.deliveries.aggregate([
        2     {"$match": {"dropoff.address.country": "FR"}}]))
```

```
In [43]: 1 print(dropoffFrance[0])
```

```
{'_id': ObjectId('6043f899e8ae4cd2eabc78b0'), 'dropoff': {'notes': 'Verificar direccion', 'place': '270 - Ayudemos de corazón',
'latlng': '43.7448016,1.3707357000000068', 'address': {'id': 'def7a335-a654-4789-9b66-2a7ddae531f', 'city': 'Haute-Garonne', 's
tate': 'Occitanie', 'country': 'FR', 'street_1': 'Square du Faur', 'street_2': None, 'zip_code': None, 'telephone': None}, 'con
tact': {'name': 'Liliana Mbareuque', 'email': '@', 'cellphone': '156-714424'}, 'open_hours': None, 'gateway_data': {}, 'id':
'3815', 'extras': {'options': []}, '_sdc_table_version': '1565122950921', 'order_id': '21b30bbf-dd43-49a2-8f07-06c95ddc8dc3',
'updated_at': '2019-08-20 13:30:57.332 UTC', 'status': 'processing', '_sdc_received_at': '2019-08-20 14:22:56.138 UTC', '_sdc_s
equence': '1566310976000000002', 'amount': 300, 'origin_id': 'ed12fd7-c085-40b4-ab7a-2f74f6b23a04', 'created_at': '2019-08-20
13:30:57.332 UTC', 'pickup': {'notes': 'Se ingresa por mitad de cuadra', 'place': '00-Banco de Alimentos - Manos Solidarias',
'latlng': '-37.9784402,-57.547047099999986', 'address': {'id': 'ed12fd7-c085-40b4-ab7a-2f74f6b23a04', 'city': 'General Pueyrre
dón', 'state': 'Buenos Aires', 'country': 'AR', 'street_1': 'Dardo Rocha 154', 'street_2': None, 'zip_code': None, 'telephone':
None}, 'contact': {'name': 'Gabriela Villafañe', 'email': 'gabriela@manos-solidarias.org.ar', 'cellphone': None, 'open_hours':
{}}, '_sdc_batched_at': '2019-08-20 14:41:42.852 UTC', '_sdc_extracted_at': '2019-08-20 14:22:55.756 UTC', 'destination_id': 'd
ef7a335-a654-4789-9b66-2a7ddae531f'}
```

- **Observaciones finales**

La exploración parcial de los datos nos ha mostrado que los mismos no están curados y que lo más recomendable es restringir lo más posible la elección de datos para una prueba de concepto.

Queda claro que, en caso que la empresa desee avanzar en el proyecto, deberá previamente realizarse un proceso de data cleansing y una gestión en el ingreso de los datos.

11 ANEXO II - Exploración de estrategia en Neo4j, Consideraciones preliminares y Análisis y selección de Algoritmos[13]

Detallamos en este apartado la investigación sobre las posibilidades de aplicar algoritmos a la información en Neo4j a los efectos de poder predecir y clasificar los diferentes Nodos en la base de datos.

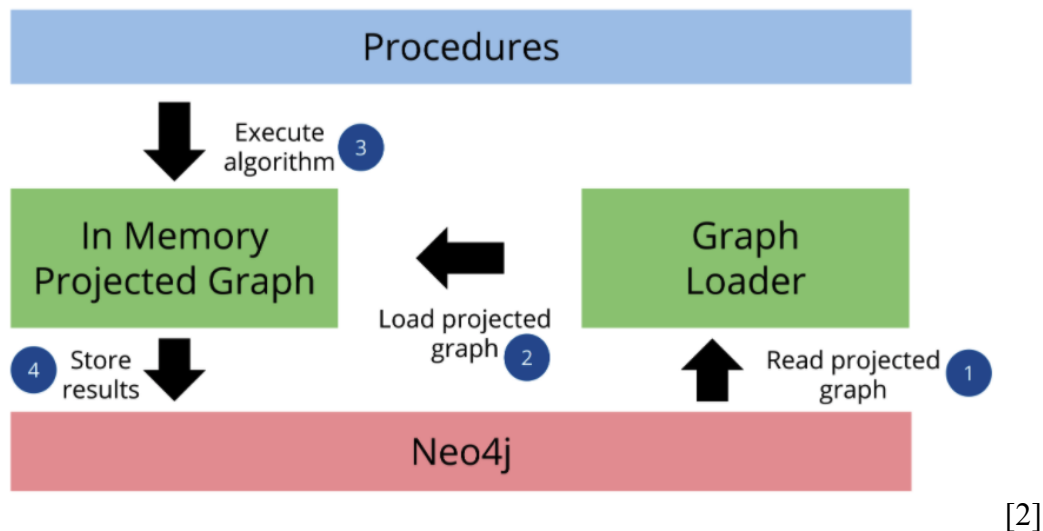
Los algoritmos de grafos se pueden utilizar para medir grafos, nodos y relaciones.

A los efectos de correr de modo eficiente los algoritmos gráficos, los mismos son ejecutados enteramente en memoria, por lo que requieren de un control de tamaño para los grafos de gran porte.

La biblioteca de GDS (Graph Data Science) se puede dividir en dos:

Desarrollo y producción

Proceso.



Formas de obtener el resultado de aplicar los algoritmos:

- **Stream** - Entrega los resultados del algoritmo como una línea de Cypher
- **Stats** - Devuelve los resultados de cálculo como números o porcentajes.
- **Mutate** - Devuelve los resultados directo a memoria.

- **Write** - Escribe los resultados directo a la base de Neo4J

- **Manejo de los grafos para Algoritmos:**

- **Graph Catalog** - Es un concepto del GDSL (Graph data science library) que permite manejar múltiples proyecciones de grafos por nombre. Luego de su utilización pueden ser fácilmente removidos por nombre.
- **Native Projection** - Nos permite proyectar un grafo de Neo4j a memoria especificando en términos de nodos y propiedades, utilizando proyección de nodos y de propiedades. Su principal beneficio es la performance.
- **Cypher Projection** - Se utiliza cuando Native Projections no tienen la potencia de expresión suficiente
- **Anonymous graphs** - Es útil para minimizar las lecturas desde Neo4j y ejecutar algoritmos con los mismos parámetros o varios algoritmos con la misma proyección gráfica.

- **Clasificación de Algoritmos:**

- **Algoritmos de Centralidad:**

- **Page Rank** - Permite medir la importancia de cada nodo en el grafo basado en el número de relaciones que recibe y la importancia de los nodos adyacentes o relacionados.
- **Betweenness Centrality** - Un modo de medir la influencia que un nodo posee en el flujo de información dentro de un grafo. Utilizado para encontrar nodos puente de una parte a otra del grafo.
- **Article Rank**- Variante de Page Rank que mide la conectividad de nodos o transitividad.
- **Closeness Centrality** - Permite encontrar nodos capaces de esparcir información de forma eficiente dentro del grafo.

- **Harmonic Centrality** - Variante del anterior diseñado para superar el problema de grafos no conectados.
- **Degree Centrality** - Mide el número de relaciones entrantes y salientes de cada nodo, ayuda a encontrar los nodos más populares.
- **Eigenvector Centrality** - Mide la conectividad entre nodos o influencia transitiva.
- **HITS** - (Hyperlink Induced Topic Search) rankea los nodos basándose en dos condiciones, un Hub score (mide el valor de las relaciones de cada nodo) y un Authority score (mide la importancia del nodo dentro de la red)

- **Algoritmos de detección de comunidades.**

Utilizados para evaluar cómo los grupos se reúnen o participan, su tendencia a unirse o partitionarse.

- **Louvain** - Detecta comunidades dentro de grandes grafos.
- **Label Propagation** - Detecta rápidamente comunidades dentro del grafo, no requiere información previa de la red.
- **Weakly Connected Components** - Utilizado por lo general como primer algoritmo para entender el estado del grafo, encuentra grupos de nodos conectados en un grafo no direccionado.
- **Triangle count** - Cuenta la cantidad de triángulos para cada nodo en el grafo.
- **Local Clustering Coefficient** - Computa un coeficiente de clustering (agrupación) de cada nodo, mide la verosimilitud de que los nodos adyacentes también estén conectados.
- **K-1 Coloring** - Asigna un color a cada nodo en el grafo optimizado para lograr dos objetivos. Que cada nodo vecino tenga un color diferente al nodo objeto, utilizar la mínima cantidad de colores.

- **Modularity Optimization** - Trata de detectar comunidades dentro del grafo basado en su modularidad, midiendo la densidad de comunicaciones dentro de cada comunidad.
- **Strongly connected Components** - Encuentra sets de nodos con máximas conexiones. Se utiliza por lo general en análisis inicial de las redes.
- **Speaker - Listener Label Propagation** - Es una variación el anterior capaz de detectar múltiples comunidades por nodo, se utiliza en forma iterativa.

○ Algoritmos de similitud

Computan la similitud entre pares de nodos utilizando métrica vectorial.

- **Node Similarity** - Compara conjuntos de nodos basándose en sus conexiones, dos nodos son considerados similares si comparten los mismos vecinos. (conocido como Jaccard similarity score)
- **K-Nearest Neighbors** - calcula el valor de distancia entre todos los pares de nodos y crea nuevas relaciones. La distancia se calcula basada en las propiedades, las relaciones existentes son ignoradas.
- **Jaccard Similarity** - Mide las similitudes entre sets de nodos. (se define como el tamaño de la intersección/ el tamaño de la unión)
- **Cosine Similarity** - Se define como el ángulo entre dos vectores n-dimensionales en un espacio n-dimensional.
- **Pearson Similarity** - Se define como la covarianza de dos vectores n-dimensionales dividido el producto de su desviación Standard.
- **Euclidean Distance** - Mide la distancia recta entre dos puntos en un espacio n-dimensional
- **Overlap Similarity** - Mide la superposición entre dos sets, se define como el tamaño de la intersección de ambos sets dividido el tamaño del más pequeño.

- **Approximate Nearest Neighbors (ANN)** - Utilizando la fórmula de Jaccard para la similitud construye un grafo para un set de nodos.

○ **Algoritmos para selección de caminos.**

- **Dijkstra Source-Target**

El algoritmo Dijkstra Shortest Path (Recorrido más corto) calcula el recorrido más corto entre nodos. Sopesa grafos con pesos (siempre positivos). Calcula desde el nodo de partida al nodo final.

- **Dijkstra's Single-Source**

Este algoritmo calcula el recorrido más corto entre un nodo de partida y todos los nodos accesibles desde éste.

- **A* (A-Star)**

A diferencia de los anteriores este algoritmo, el próximo nodo a buscar no solo lo estima basado en distancia, toma en cuenta la distancia estimada junto con una función heurística, el algoritmo itera hasta encontrar el costo combinado más bajo.

- **Yen's Algorithm**

El algoritmo Yen's Shortest Path, calcula un número k de recorridos más cortos entre dos nodos.

- **Minimum Weight Spanning Tree**

El MST o Minimum Weight Spanning Tree, parte de un nodo dado y encuentra todos los nodos accesibles y todas las relaciones que los conectan con el menor peso posible.

- **Single Source Shortest Path**

El Single Source Shortest Path (SSSP) calcula el mínimo recorrido con peso desde un nodo a todos los demás nodos del grafo.

- **All Pairs Shortest Path**

APSP calcula el mínimo recorrido con pesos entre todos los nodos del grafo, el algoritmo está optimizado para ser más rápido que el SSSP

- **Random Walk**

Comenzamos en un nodo y aleatoriamente o basados en una distribución de probabilidades navegamos a un nodo vecino, y así continúa guardando los recorridos en una lista (caminata de borracho)

- **Breadth First Search**

Es un algoritmo que dado un nodo de inicio recorre o atraviesa todo el grafo buscando nodos basado en distancias crecientes.

- **Depth First Search**

Es un algoritmo que atraviesa todo el grafo, comenzando en un nodo recorre totalmente cada rama para luego regresar.

- **Algoritmos de predicción de enlaces**

Sirven para determinar la cercanía entre pares de nodos.

- **Adamic Adar** - Fórmula para predecir la cercanía entre dos nodos basada en sus contactos compartidos.
- **Common Neighbors** - Fórmula que captura la idea que dos nodos con un nodo en común son más posibles de relacionarse que aquellos sin nada en común.
- **Preferential Attachment** - Medida utilizada para computar la cercanía de nodos basada en sus contactos en común.
- **Resource Allocation** - Otro algoritmo para computar la cercanía entre nodos basado en sus vecinos compartidos.
- **Same Community** - Luego de determinadas las comunidades utilizando "Community detection algorithms", este nos permite determinar si dos nodos pertenecen a la misma comunidad o no.

- **Total Neighbors** - computa la cercanía entre nodos, basado en el número de relaciones únicas que cada nodo tiene, incrementa la posibilidad de recibir nuevos contactos.

○ **Algoritmos de inserción de nodos.**

Este tipo de algoritmos utiliza y computa la representación de nodos en forma vectorial en el grafo. dichos vectores son llamados embedding, son utilizados para Machine Learning.

- **FastRP** - Fast Random Projection basado en el Johnson-Lindenstrauss lema, según el cual se pueden proyectar n vectores de dimensión arbitraria en un espacio n -dimensional y preservar la paridad de distancias entre los puntos. Nodos con similares vecinos se les asigna similares vectores. Comienza asignando vectores en forma aleatoria y luego itera
- **GraphSAGE** - Algoritmo inductivo para computar embeddings (vectores) utiliza un sistema de propiedades de nodos para generar los vectores en nodos o grafos no vistos. En lugar de entrenar en nodos individuales el algoritmo aprende de los nodos vecinos.
- **Node2Vec** - Computar una representación vectorial de cada nodo basado en random walk, entrena una capa de red neuronal para predecir la posibilidad que aparezca un nodo en la ocurrencia de otro nodo.

○ **Modelos de aprendizaje automático**

Permite procedimientos en Neo4j para entrenar modelos supervisados. Esto nos ayuda a realizar predicciones sobre el grafo.

Los modelos incluidos en GDS Library (Graph Data System Library) son:

- **Node Classification** - Entrena un modelo para evaluar en donde un nodo va a pertenecer, basado en aprendizaje y las propiedades del nodo.
- Permite separar el grafo en Train y Test ... etc.
- **Link Prediction** - Similar al anterior, entrena un modelo para aprender entre pares de nodos en un grafo, donde deberían existir relaciones, se puede interpretar como crear un modelo para saber donde deben existir relaciones.

○ **Procedimientos auxiliares.**

- **Graph Generation** - Generador de grafos para propósitos de test y entrenamiento.
- **Collapse Path** - Generador de relaciones entre nodos de principio y fin para ser utilizados en otros algoritmos.
- **One Hot Encoding** - Utilizado para convertir datos categóricos en formato numérico para ser utilizado en bibliotecas y algoritmos de machine learning.
- **Split Relationships** - Es un algoritmo utilizado para preparar el grafo en train y test, para ser utilizado en procedimientos de Machine Learning.

Luego del anterior resumen de los principales algoritmos que ofrece Neo4j podemos comenzar a ver cómo aplicarlos a nuestro proyecto.

Una vez que el grafo de Nilus se encuentre totalmente cargado y con por lo menos una carga de 1 año de datos a los efectos de contar con las 4 temporadas.

12 ANEXO III - Middleware en Python

• MongoDB a Neo - Transformar y Exportar los Datos

En esta sección veremos el proceso de transformación de los datos almacenados en MongoDB a fin de poder exportarlos a Neo4j.

```
[2]: #conda install -c conda-forge neo4j-python-driver
     #!pip install osm-runner
```

```
[20]: #import libraries

import json
from pymongo import MongoClient
import dns
```

En primera instancia nos conectamos al servidor MongoDB en la nube, y con dicho cliente nos conectamos a la base de datos Nilus_Test.

```
[21]: #creamos un cliente en el servidor de la nube descrito en el documento principal

nilusClient = MongoClient("mongodb+srv://PYORT-adm1:3HkrANGmREZeggw@cluster0.
    →uxk4d.mongodb.net/Nilust_Test.Deliveries?retryWrites=true&w=majority")

#nos conectamos con la base Nilus_Test
db = nilusClient["Nilust_Test"]
```

```
[22]: #chequeamos que la base esté en el servidor

print(nilusClient.list_database_names())
```

```
['Nilust_Test', 'sample_airbnb', 'sample_analytics', 'sample_geospatial',
'sample_mflix', 'sample_restaurants', 'sample_supplies', 'sample_training',
'sample_weatherdata', 'admin', 'local']
```

Seguidamente realizamos un procesamiento, transformación y adecuación de los datos que tenemos almacenados en MongoDB ya cómo los vamos a impactar en neo4j. Los pasos:

- Filtramos los datos pertenecientes a la ciudad de Rosario en la Provincia de Santa Fe, Argentina
- Elegimos los campos que vamos a utilizar
- Separamos los datos de latitud y longitud, que venían en un solo campo
- Cargamos todos los documentos de MongoDB en una lista para ser insertada en neo4j la cual sería una estructura suficiente para generar el nodo.

```
[4]: #from collections import OrderedDict

dropoffList = list(db.deliveries.aggregate([
    { "$match": { "dropoff.address.city": "Rosario" } },
    { "$project": { "dropoff.latlng":1,"dropoff.place": 1,"dropoff.address.id":
↳1,"dropoff.address.street_1":1 ,"_id":0} },
    #{ "$sort": OrderedDict([("dropoff.address.id",1)])},
    {"$group": {
        "_id" : "$dropoff.address.id",
        "place" : { "$first": "$dropoff.place" },
        "latlng" : { "$first": "$dropoff.latlng" },
        "street":{"$first":"$dropoff.address.street_1"}
    }
    },
    {
        "$addFields": {
            "commaPos": { "$indexOfBytes": ["$latlng",""] },
            "longPos":{ "$sum": [ "$commaPos",1] }
        }
    },
    {
        "$addFields": {
            "latitude": { "$substr": [ "$latlng", 0, "$commaPos" ] },
            "longitude": { "$substr": [ "$latlng",{ "$sum": [ "$commaPos",1] }],
↳{"$strLenBytes": "$latlng" } ] }
            # "longitude": { "$longPos" }

        }
    },
    {
        "$project": {
            "commaPos": 0
        }
    }
])

len(dropoffList)
```

```
[30]: for place in dropoffList[:5]:
        print(place)

{'_id': 'fcf37b37-3c5a-4d9d-810f-cecae751e760', 'place': '259 - Cooperativa Flor Del Ceibo', 'latlng': '-32.9197,-60.6923', 'street': 'Juan José Paso 2067', 'longPos': 1, 'latitude': '-32.9197', 'longitude': '-60.6923'}
{'_id': '9cf33d1e-ffb3-4ca6-a1a8-60f2e6961340', 'place': '150 - Asociación Civil Santa Catalina', 'latlng': '-33.0171478,-60.662299899999994', 'street': '5 de Agosto 2102', 'longPos': 1, 'latitude': '-33.0171478', 'longitude': '-60.662299899999994'}
{'_id': '130586c8-f371-401a-887e-f759feaba286', 'place': '56 - Cielos Abiertos', 'latlng': '-32.9526382,-60.712323200000014', 'street': 'Av. Provincias Unidas 2050', 'longPos': 1, 'latitude': '-32.9526382', 'longitude': '-60.712323200000014'}
{'_id': '36fcd89b-723b-4029-b5d8-668b1ee55e45', 'place': '390 - Parroquia Nuestra Señora del Valle (Centro de Vida) ', 'latlng': '-32.9932693,-60.660611399999999', 'street': 'Dorrego 6178', 'longPos': 1, 'latitude': '-32.9932693', 'longitude': '-60.660611399999999'}
{'_id': '53457c6d-4266-4b3c-b820-0138f788c711', 'place': '135 - Asociación Civil Centro Diplomático Rosario ', 'latlng': '-32.9937538,-60.630900300000001', 'street': 'Cucha Cucha 4820', 'longPos': 1, 'latitude': '-32.9937538', 'longitude': '-60.630900300000001'}
```

Acto seguido impactamos sobre neo4j, conectándonos y trabajando sobre la base de datos montada y generada en la nube.

```
[6]: from neo4j import GraphDatabase

[ ]: uri = "neo4j+s://4d7daebf.databases.neo4j.io"
      driver = GraphDatabase.driver(uri, auth=("neo4j",
      ↪ "igcpilwMFZD3DEtDmqdtqLmcdeNeemvu-UOTDZjkOdW"))
```

Aquí realizaremos el proceso de inserción de los nodos en neo4j a partir de los datos extraídos de MongoDB.

```
[38]: #generamos la función para insertar los nodos en neo4j

def insert_nodes_of(xt,list):

    for dropPlace in dropoffList:
        idAdress=dropPlace["_id"]
        place=dropPlace["place"]
        latitude=dropPlace["latitude"]
        longitude=dropPlace["longitude"]
        street=dropPlace["street"]
```

```

        query="CREATE (n:Dropoff {idAddress:'"+idAddress+"', place:'"+place+"',
        ↪latitude:'"+latitude+"', longitude:'"+longitude+"', street:'"+street+"'})"
        print(query)
        xt.run(query)

with driver.session() as session:

    session.write_transaction(insert_nodes_of,dropoffList)

driver.close()

```

```

CREATE (n:Dropoff {idAddress:'fcf37b37-3c5a-4d9d-810f-cecae751e760', place:'259 -
Cooperativa Flor Del Ceibo', latitude:-32.9197, longitude:-60.6923, street:'Juan
José Paso 2067'})
CREATE (n:Dropoff {idAddress:'9cf33d1e-ffb3-4ca6-a1a8-60f2e6961340', place:'150 -
Asociación Civil Santa Catalina', latitude:-33.0171478,
longitude:-60.662299899999994, street:'5 de Agosto 2102'})
CREATE (n:Dropoff {idAddress:'130586c8-f371-401a-887e-f759feaba286', place:'56 -
Cielos Abiertos', latitude:-32.9526382, longitude:-60.712323200000014,
street:'Av. Provincias Unidas 2050'})
CREATE (n:Dropoff {idAddress:'36fcd89b-723b-4029-b5d8-668b1ee55e45', place:'390 -
Parroquia Nuestra Señora del Valle (Centro de Vida) ', latitude:-32.9932693,
longitude:-60.660611399999999, street:'Dorrego 6178'})
CREATE (n:Dropoff {idAddress:'53457c6d-4266-4b3c-b820-0138f788c711', place:'135 -
Asociación Civil Centro Diplomático Rosario ', latitude:-32.9937538,
longitude:-60.630900300000001, street:'Cucha Cucha 4820'})
CREATE (n:Dropoff {idAddress:'a4b0ccef-71d2-4180-a996-f4edbce09428', place:'21 -
Comedor la morena', latitude:-32.9806, longitude:-60.6965, street:'Comandos 602
4384'})
CREATE (n:Dropoff {idAddress:'84e89679-761a-413f-bb6a-83883fbabb8e', place:'37 -
Asociación Nazareth ', latitude:-32.9319945, longitude:-60.658044000000002,
street:'Güemes 2735'})
CREATE (n:Dropoff {idAddress:'fe8c5dc8-6234-4a28-81ec-eff67c620dcc', place:'269 -
Tupac Amaru', latitude:-32.9289625, longitude:-60.707371099999999,
street:'Casilda 6202'})
CREATE (n:Dropoff {idAddress:'927c7f07-3a9f-4d32-a8eb-146c2c74123f', place:'119 -
Cooperativa el progreso', latitude:-32.9170077, longitude:-60.708861299999997,
street:'Campbell 999'})
CREATE (n:Dropoff {idAddress:'d6051740-ffe5-45ab-b47e-88337043ba3a', place:'248 -
FON-BEC', latitude:-32.93193, longitude:-60.706045000000002, street:'Garzón
351'})
CREATE (n:Dropoff {idAddress:'bf261b56-def0-4920-aaa1-9744c49530dc', place:'184 -
Capilla Virgen de la Granada', latitude:-33.0119113, longitude:-60.6568709,
street:'España 6417'})
CREATE (n:Dropoff {idAddress:'0bd6cd71-56bc-48a2-a522-0ae541c38f57', place:'311 -
Asociación Civil Vecinos Olvidados', latitude:-33.0176299,
longitude:-60.652371499999996, street:'San Martín 6921'})
CREATE (n:Dropoff {idAddress:'746a9155-b789-47a9-b0b6-1046725a9721', place:'194 -

```

Ahora generamos una serie de rutas arbitrarias entre nodos, de modo de tener un juego de datos de trabajo. A todas les asignamos un peso wgt arbitrario, que se irá modificando a medida que sea necesario. Realizamos dos funciones distintas, solo para poder ver rutas alternativas.

```
[41]: #genero relaciones básicas entre nodo

def insert_routes_of(xt,list):

    for i, dropPlace in enumerate(dropoffList[:10]):

        dropNext=dropoffList[i+1]
        idAddress=dropPlace["_id"]
        latitude=dropPlace["latitude"]
        longitude=dropPlace["longitude"]

        street=dropPlace["street"]
        idAddressN=dropNext["_id"]
        latitudeN=dropNext["latitude"]
        longitudeN=dropNext["longitude"]
        query="MATCH (a:Dropoff), (b:Dropoff) WHERE (a.idAddress='"+idAddress+"')
→and (b.idAddress='"+idAddressN+"') CREATE (a)-[r:ROAD {wgt:20}]->(b)"
        #query="MATCH (a:Dropoff), (b:Dropoff) WHERE (a.latitude='"+latitude+"
→and a.longitude='"+longitude+"') and (b.latitude='"+latitudeN+" and b.
→longitude='"+longitudeN+"') CREATE (a)-[r:ROAD {distance:100, wgt:1}]->(b)"
        print(query)
        xt.run(query)

#genero otro juego de relaciones secundarias

def insert_other (xt, list):

    drop=dropoffList[0]
    drop5th=dropoffList[4]
    drop9th=dropoffList[8]
    idAddress=drop["_id"]
    idAddress5=drop5th["_id"]
    idAddress9=drop9th["_id"]
    query1="MATCH (a:Dropoff), (b:Dropoff) WHERE (a.idAddress='"+idAddress+"') and
→(b.idAddress='"+idAddress5+"') CREATE (a)-[r:ROAD {wgt:3,date_events_reported:
→['2021-03-05','2021-03-08'],event_description:['Piquete en la zona','Calle en
→refaccion']}]>(b)"
    query2="MATCH (a:Dropoff), (b:Dropoff) WHERE (a.idAddress='"+idAddress+"') and
→(b.idAddress='"+idAddress9+"') CREATE (a)-[r:ROAD {wgt:7,date_events_reported:
→['2021-03-02','2021-03-09'],event_description:['Piquete en la
→zona','Disturbios']}]>(b)"
    query3="MATCH (a:Dropoff), (b:Dropoff) WHERE (a.idAddress='"+idAddress5+"')
→and (b.idAddress='"+idAddress9+"') CREATE (a)-[r:ROAD {wgt:
→6,date_events_reported:['2021-03-02','2021-03-09'],event_description:['Piquete
→en la zona','Disturbios']}]>(b)"
    xt.run(query1)
    xt.run(query2)
    xt.run(query3)

#llamo a ambas funciones

with driver.session() as session:
    session.write_transaction(insert_routes_of,dropoffList)
    session.write_transaction(insert_other,dropoffList)

driver.close()
```

```

MATCH (a:Dropoff), (b:Dropoff) WHERE
(a.idAddress='fcf37b37-3c5a-4d9d-810f-cecae751e760') and
(b.idAddress='9cf33d1e-ffb3-4ca6-a1a8-60f2e6961340') CREATE (a)-[r:ROAD
{wgt:20}]->(b)
MATCH (a:Dropoff), (b:Dropoff) WHERE
(a.idAddress='9cf33d1e-ffb3-4ca6-a1a8-60f2e6961340') and
(b.idAddress='130586c8-f371-401a-887e-f759feaba286') CREATE (a)-[r:ROAD
{wgt:20}]->(b)
MATCH (a:Dropoff), (b:Dropoff) WHERE
(a.idAddress='130586c8-f371-401a-887e-f759feaba286') and
(b.idAddress='36fcd89b-723b-4029-b5d8-668b1ee55e45') CREATE (a)-[r:ROAD
{wgt:20}]->(b)
MATCH (a:Dropoff), (b:Dropoff) WHERE
(a.idAddress='36fcd89b-723b-4029-b5d8-668b1ee55e45') and
(b.idAddress='53457c6d-4266-4b3c-b820-0138f788c711') CREATE (a)-[r:ROAD
{wgt:20}]->(b)
MATCH (a:Dropoff), (b:Dropoff) WHERE
(a.idAddress='53457c6d-4266-4b3c-b820-0138f788c711') and
(b.idAddress='a4b0cceff-71d2-4180-a996-f4edbce09428') CREATE (a)-[r:ROAD
{wgt:20}]->(b)
MATCH (a:Dropoff), (b:Dropoff) WHERE
(a.idAddress='a4b0cceff-71d2-4180-a996-f4edbce09428') and
(b.idAddress='84e89679-761a-413f-bb6a-83883fbabb8e') CREATE (a)-[r:ROAD
{wgt:20}]->(b)
MATCH (a:Dropoff), (b:Dropoff) WHERE
(a.idAddress='84e89679-761a-413f-bb6a-83883fbabb8e') and
(b.idAddress='fe8c5dc8-6234-4a28-81ec-eff67c620dcc') CREATE (a)-[r:ROAD
{wgt:20}]->(b)
MATCH (a:Dropoff), (b:Dropoff) WHERE (a.idAddress='fe8c5dc8-6234-4a28-81ec-
eff67c620dcc') and (b.idAddress='927c7f07-3a9f-4d32-a8eb-146c2c74123f') CREATE
(a)-[r:ROAD {wgt:20}]->(b)
MATCH (a:Dropoff), (b:Dropoff) WHERE
(a.idAddress='927c7f07-3a9f-4d32-a8eb-146c2c74123f') and
(b.idAddress='d6051740-ffe5-45ab-b47e-88337043ba3a') CREATE (a)-[r:ROAD
{wgt:20}]->(b)
MATCH (a:Dropoff), (b:Dropoff) WHERE
(a.idAddress='d6051740-ffe5-45ab-b47e-88337043ba3a') and
(b.idAddress='bf261b56-def0-4920-aaa1-9744c49530dc') CREATE (a)-[r:ROAD
{wgt:20}]->(b)

```

```
[ ]: print(dropoffList[1])
```

- **Rutas en OpenStreetMap**
- **Recuperación y marcado de rutas**

En el siguiente script implementaremos algunos de los pasos mostrados en el documento central, particularmente la recuperación de las rutas entre nodos almacenadas en neo4j, así como la resolución práctica de la misma a través de los servicios de Open Street Maps. El lenguaje en que realizaremos la intermediación es python.

Para conectarnos con Open Street Maps desde python utilizaremos la API ArcGis, la cual está ampliamente documentada y cuenta con un conjunto de librerías y servicios amplia -en la

bibliografía se incluyen las citas pertinentes. Aquí utilizaremos solamente aquellos servicios que están en la capa abierta para desarrolladores, aunque haremos referencia también a servicios pagos que presentan y son pertinentes para el caso de estudio.

Por otro lado, nos conectaremos con neo4j, en donde ya tenemos cargados los nodos y relaciones entre ellos. Desde aquí invocamos también funciones y servicios de la librería apoc, particularmente aquellos algoritmos de ruteo básico -nuevamente, en la bibliografía se encuentran las citas pertinentes. Aquí sucede lo mismo que lo comentado para ArcGIS, utilizaremos la capa de funcionalidad básica abierta en AURA.

Dejamos comentadas las instalaciones básicas que se deben realizar para correr el código que está más abajo.

```
#Dejamos aquí Las Librerías imprescindibles para correr el código., tal y como Las instalamos desde Anaconda

#install Libraries
#!pip install osm-runner
#!pip install arcgis
#!pip install datetime
#!conda install -c esri arcpy
```

- **Librerías**

Primeramente, importamos las librerías de Open Street Maps y Neo4j que necesitaremos para correr el código, así como otras que serán utilizadas para fines específicos.

```
#imports necesarios para correr Los ejempLos

import time

from osm_runner import Runner # pip install osm-runner
import json
import pandas as pd
import arcgis
from neo4j import GraphDatabase

from copy import deepcopy
from arcgis.features import FeatureLayer, FeatureLayerCollection, GeoAccessor, GeoSeries Accessor
from arcgis.geoenrichment import enrich
from arcgis import dissolveboundaries
from arcgis.geometry import project
from arcgis.geometry import SpatialReference
from arcgis.gis import GIS
import arcgis.network as network
from arcgis import geocoding
from arcgis.features import Feature, FeatureSet

import getpass
import datetime
from copy import deepcopy
```

• Conexión a Servicios

Nos conectamos a los servicios de Neo4j y ArcGIS que hemos montado en la nube.

```
# Connect
user_name = 'icarbonell31'
password = 'PlayaVerde-2021'
gis = GIS('https://www.arcgis.com', user_name, password)
#gis = GIS('https://www.arcgis.com', user_name, password, verify_cert=False, set_active=True)
```

In [250]:

```
uri = "neo4j+s://4d7daebf.databases.neo4j.io"
driver = GraphDatabase.driver(uri, auth=("neo4j", "igcpilwMFZD3DEtDmqdtqLmcdeNeemvu-UOTDZjk0dw"))
```

Nos conectamos a algunos servicios de “arcgis” que utilizaremos, particularmente el servicio de ruteo básico, confirmando que la url sea la correcta.

```
route_service_url = gis.properties.helperservices.route.url
route_service_url
```

Out[227]: 'https://route.arcgis.com/arcgis/rest/services/WorId/Route/NA Server/Route World'

In [251]:

```
#servicio reLevante para eL momento de trazar La ruta especifica
route_service = network.RouteLayer(route_service_url, gis=gis)
route_service
```

Out[251]:

<RouteLayer url:"https://route.arcgis.com/arcgis/rest/services/World/Route/NA Server/Route World">

Solo a modo exploratorio mostramos los servicios que tenemos expuestos. En lo subsiguiente debe tenerse en cuenta un detalle, en estos servicios la X se corresponde con la longitud y la Y se corresponde con la latitud. Por tanto, en nuestra tradición de nombrar las coordenadas como latitud/longitud debemos ver que serán pares Y/X. Por otro lado, tener en cuenta también, que el valor máximo representa los puntos Norte y Este, así como los valores mínimos representan los puntos Sur y Oeste.

- **Recuperación de Rutas de Neo4j - Trazado de Ruta Simple**

Como método sencillo inicial, probaremos recuperar una ruta entre dos nodos que existe.

Como establecimos en el documento, esa conexión entre dos nodos puede existir distintas instancias de delivery realizadas en el pasado, por lo cual esos dos nodos pueden aparecer en diversas rutas. Como aproximación para definir con cuál ruta quedarnos, utilizaremos el algoritmo “all Simple Paths” de apoc, el cual devuelve todas las rutas posibles, y nos quedaremos con la primera, la más corta. A la consulta a neo4j le pasamos la instrucción de ordenarlas por el largo de la ruta.

```
#query_neo="MATCH (from:Dropoff {Latitude:-32.9932693; Longitude:-60.66061139999999})> (to:Dropoff
{Latitude:-32.9937538; Longitude:-60.63090030000001}) CALL apoc.algo.dijkstra(from, to, 'ROAD', 'd')
yield path as path; weight as weight RETURN path" query_neo="MATCH (from:Dropoff
{latitude:-32.9932693,longitude:-60.66061139999999}),(to:Dropoff
{latitude:-32.9937538,longitude:-60.63090030000001}) CALL apoc.algo.allSimplePaths(from, to, 'ROAD', 5)
YIELD path RETURN path ORDER BY length(path)"
#query_neo="MATCH (from:Dropoff {Latitude:-32.9932693; Longitude:-60.66061139999999}); (to:Dropoff
{Latitude:-32.9937538; Longitude:-60.63090030000001}) CALL apoc.algo.aStar(from, to, 'ROAD', 'wgt';
'Latitude', 'Longitude') YIELD path; weight RETURN path; weight"

with driver.session() as session{
    nodes= session.run(query_neo)
    result_nodes = [record for record in nodes.data()]

result_nodes

driver.close()
```

Cargamos en un panda para visualizar, a grandes rasgos, cuántas rutas se obtuvieron.

In [258]:

```
df_allSimplePaths = pd.DataFrame(result_nodes)
df_allSimplePaths
```

Out[256]:

	path
0	[{'place': '390 - Parroquia Nuestra Senora del Valle (Centro de Vida)', 'idAdress':'36fcd89b-...
1	[{'place': '390 - Parroquia Nuestra Senora del Valle (Centro de Vida)', 'idAdress':'36fcd89b-...
2	[{'place': '390 - Parroquia Nuestra Senora del Valle (Centro de Vida)', 'idAdress':'36fcd89b-...

Vemos que tenemos tres caminos posibles, queremos verificar la naturaleza de los mismos.

Como el esquema es NODO - RUTA - NODO, asumimos que 0 y los lugares pares pertenecen a los nodos, mirando los primeros nodos e investigando el largo de los mismos, solo a modo exploratorio para ver los nodos establecidos en cada ruta. Distinguiremos, entonces, los "path" de los "pasos" de cada una de ellas. Allí vemos que, efectivamente, la primera es la más corta de las rutas que tenemos.

```

x=0
y=0
for i in result_nodes:
    y=0
    x+=1
    print(len(i['path']))
    print("path ",x)
    while y < len(i['path']): print("paso ",y/2)
        print(i['path'][y]['latitude']) print(i['path'][y]['longitude']) y += 2

3
path 1 paso 0.0 -32.9932693 -60.660611399999999 paso 1.0
-32.9937538 -60.630900300000001
9
path 2 paso 0.0 -32.9932693 -60.660611399999999 paso 1.0
-32.9526382 -60.712323200000014 paso 2.0
-33.0171478 -60.662299899999994 paso 3.0
-32.9197
-60.6923 paso 4.0 -32.9937538 -60.630900300000001
11
path 3 paso 0.0 -32.9932693 -60.660611399999999 paso 1.0
-32.9526382 -60.712323200000014 paso 2.0
-33.0171478 -60.662299899999994 paso 3.0
-32.9197
-60.6923 paso 4.0 -32.9170077 -60.708861299999997 paso 5.0
-32.9937538 -60.630900300000001

```

Si bien para este paso no tiene mucho sentido -ya que tan solo hay dos puntos para unir-, de todos modos como forma de ejemplificar, utilizaremos el algoritmo aStar de la librería apoc, para devolver una ruta optimizada

```

query_neo="MATCH (from:Dropoff {latitude:-32.9932693,longitude:-60.660611399999999}),(to:Dropoff
{latitude:-32.9937538,longitude:-60.630900300000001}) CALL apoc.algo.aStar(fro m, to, 'ROAD','wgtlatitude',
'longitude') YIELD path, weight RETURN path, weight" with driver.session() as session:

    nodes= session.run(query_neo)
    result_nodes = [record for record in nodes.data()]

result_nodes

driver.close()

```

El algoritmo “aStar” nos devuelve el peso de la ruta, factor que indica un grado de dificultad que la misma ha tenido de acuerdo con distintas variables especificadas en el documento.

```
df_aStar = pd.DataFrame(result_nodes)
df_aStar
```

Out[263]:

	path	weight
n	[[{'place': '390 - Parroquia Nuestra Senora del Valle (Centro de Vida)', 'idAddress':	..
u	'36fcd89b-...	10

Finalmente, nos conectamos a Open Street Maps para obtener la ruta específica, paso por paso y con tiempos estimados. Para dicho fin utilizamos la función solve de la capa de enrutamiento “route layer”. Una serie de precisiones son importantes aquí, que para el caso simple no son tan relevantes, pero si lo son para rutas complejas y para la evaluación posterior de las mismas.

Primero, el método solve nos permite pasar parámetros de reordenamiento de nodos, pero para ser utilizados se debe tener contratada la capa superior de servicios.

Por otro lado, la capa de Network Analysis -no accesible para la cuenta general de desarrollador- tiene implementados los métodos “find_route” y “solve_vehicle_routing_problem”, los cuales incluyen los aspectos más relevantes para la resolution de este tipo de casos. Más adelante retomaremos este punto.

Por último, incorporamos el parámetro start_time para manejar los tiempos, hecho que una aplicación hará. Aca, como parámetro de referencia, se cargará con el valor en el tiempo de ejecución, pero obviamente recibirá ese parámetro durante el trayecto.

```
#carga un tiempo de inicio y coordenadas de pruebaj cargo La Libreria datetime startTime =
int(datetime.datetime.utcnow().timestamp() * 1000)
tripStops = ""-60.66061139999999j-32.9932693;-60.63090030000001,-32.9937538""

route_layer = network.RouteLayer(route_service_url, gis=gis)

result = route_layer.solve(stops=tripStops,
                           directions_language='es-AR',
                           return_directions=True, return_routes=True,
                           return_stops=True,
                           return_barriers=False, return_polygon_barriers=False,
                           return_polyline_barriers=False,
                           start_time=startTime,
                           start_time_is_utc=True)

travel_time = result['routes'][0]['features'][0]['attributes']['Total_TravelTime'] print("El tiempo total de viaje es
de {0:.2f} min".format(travel_time))
```

El tiempo total de viaje es de 6.75 min

Creamos una tabla sobre la ruta detallada paso a paso, en la cual se incluye la información sobre la misma, particularmente las calles y la distancia y duración de viaje para cada segmento.

```
records = []

#genero e inicio Las variables donde voy a cargar los indicadores en cada iteración

travel_time, time_counter = 0, 0
distance, distance_counter = 0, 0

#recorro el resultado obtenido cargo en el objeto record
for i in result['directions'][0]['features']:
    time_of_day = datetime.datetime.fromtimestamp(i['attributes']['arriveTimeUTC'] / 1000).strftime('%H:%M:%S')
    time_counter = i['attributes']['time']
    distance_counter = i['attributes']['length']
    travel_time += time_counter
    distance += distance_counter
    records.append((time_of_day, i['attributes']['text'],
                    round(travel_time, 2), round(distance, 2)))

from IPython.display import HTML

#cargo el panda
pd.set_option('display.max_colwidth', 100)
df = pd.DataFrame.from_records(records, index=[i for i in range(1, len(records) + 1)], columns=['Hora',
                                                'Indicacion',
                                                'Duracion (min)', 'Distancia (km)'])
```

```
HTML(df.to_html(index=False))
```

```
HTML(df.to_html(index=False))
```

Out[276]:

Hora	Indicacion	Duracion (min)	Distancia (km)
21:52:42	Saiga desde Location 1	0.00	0.00
21:52:42	Diríjase al este por Bahía Juárez hacia Melian	0.99	0.23
21:53:42	Gire a la izquierda por Presidente Roca	1.58	0.38
21:54:17	En el semáforo, gire a la derecha en Avenida Uriburu	4.57	1.53
21:57:17	En el semáforo, gire a la derecha en Estación de Israel	4.94	1.68
21:57:39	Siga adelante por J. María Gutiérrez	5.21	1.79
21:57:55	Siga adelante por Lamadrid	5.27	1.82
21:57:58	Gire a la izquierda por Esteban de Luca	6.53	2.16
21:59:14	Gire a la izquierda por Cucha Cucha	6.75	2.21
21:59:27	Ha llegado a Location 2, que se encuentra a la izquierda	6.75	2.21

Finalmente, podemos ver el recorrido trazado en el mapa específico.

```
my_map = gis.map('Rosario, Santa Fe', zoomlevel=12)

my_map.clear_graphics()
stop_count = result['routes']['features'][0]['attributes']['StopCount'] travel_time =
result['routes']['features'][0]['attributes']['Total_TravelTime'] distance =
result['routes']['features'][0]['attributes']['Total_Kilometers']

data = [('Cantidad de Paradas', stop_count),
        ('Tiempo Total de Viaje', '{0:.2f} min'.format(travel_time)),
        ('Distancia Recorrida', '{0:.2f} km'.format(distance))]

df = pd.DataFrame.from_records(data)
styles = [
    dict(selector="td", props=[("padding", "2px")]),
    dict(selector='.row_heading, .blank', props=[("display", 'none;')]), dict(selector='.col_heading, .blank',
    props=[("display", 'none;')])

symbol = {
    "type": "esriSLS",
    "style": "esriSLSolid",
    "color": [128,0,128,90],
    "width": 4
}

popup_route = {"title": "Ruta",
               "content": df.style.set_table_styles(styles).render()}
popup_stop = {"title": "Stop {} ",
              "content": df.style.set_table_styles(styles).render()}

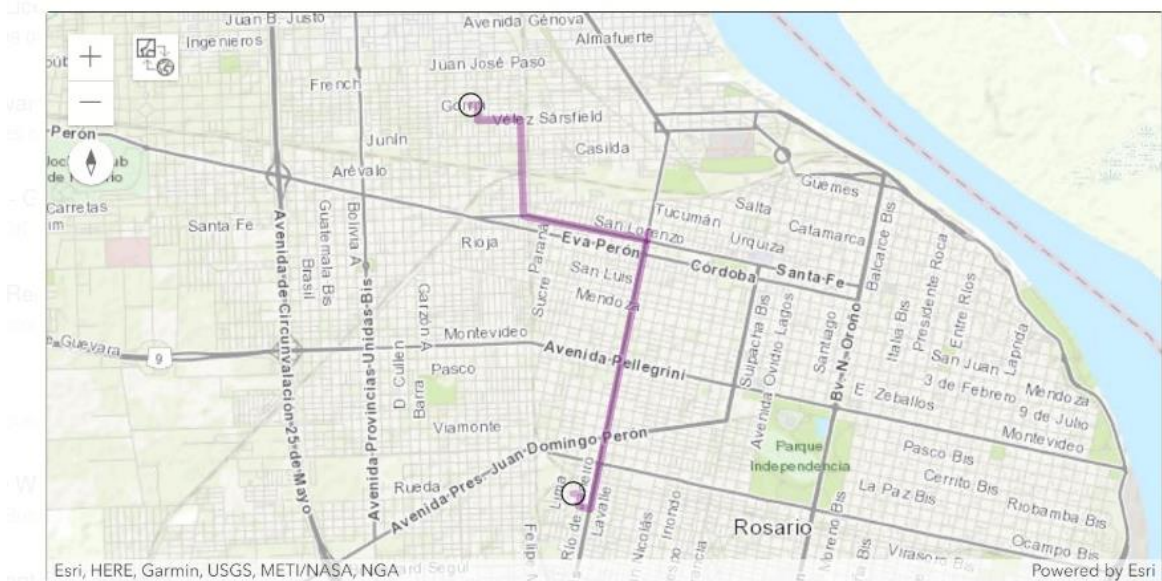
my_map.draw(result['routes']['features'][0]['geometry'], popup_route, symbol)

for stop in result['stops']['features']:
    address = geocoding.reverse_geocode(stop['geometry'])['address']['Match_addr'] print(address)
    my_map.draw(stop['geometry'],
                {"title": "Stop {}".format(stop['attributes']['Sequence']),
                 "content": address})

my_map.zoom = 12

my_map
```

Bahia Juarez 1852-1900, Rosario, S2013, Santa Fe Cucha Cucha 4802-4900, Rosario, S2013, Santa Fe



• Rutas Complejas

Si bien el caso simple puede presentarse, lo más frecuente es que se presente el caso de una ruta con 2 o más puntos de entrega. Como promedio vimos que eran 3 o 4 puntos de entrega. Cargamos algunos datos manualmente para ejemplificar el caso.

```
In [673]: 1 #genero un conjunto de datos con las direcciones, luego hacerlo trayendo los nodos de neo
2 dfTrip = pd.DataFrame(columns=["Start_X", "Start_Y", "Start_Address", "StartDepotName", "End_X", "End_Y", "EndDepotName", "Name"])
3
4 dfTrip.at[0, 'Start_Y'] = '-60.66061139999999'
5 dfTrip.at[0, 'Start_X'] = '-32.9932693'
6 dfTrip.at[0, 'Start_Address'] = 'Dorrego 6178'
7 dfTrip.at[0, 'StartDepotName'] = '56 - Cielos Abiertos'
8 dfTrip.at[0, 'End_Y'] = '-60.63090030000001'
9 dfTrip.at[0, 'End_X'] = '-32.9937538'
10 dfTrip.at[0, 'EndDepotName'] = '390 - Parroquia Nuestra Señora del Valle (Centro de Vida)'
11 dfTrip.at[0, 'Name'] = 'Route 1'
12
13 dfTrip.at[1, 'Start_Y'] = '-60.63090030000001'
14 dfTrip.at[1, 'Start_X'] = '-32.9937538'
15 dfTrip.at[1, 'Start_Address'] = 'Cucha Cucha 4820'
16 dfTrip.at[1, 'StartDepotName'] = '390 - Parroquia Nuestra Señora del Valle (Centro de Vida)'
17 dfTrip.at[1, 'End_Y'] = '-60.708861299999997'
18 dfTrip.at[1, 'End_X'] = '-32.9170077'
19 dfTrip.at[1, 'EndDepotName'] = '135 - Asociación Civil Centro Diplomático Rosario'
20 dfTrip.at[1, 'Name'] = 'Route 2'
21
22 dfTrip.at[2, 'Start_Y'] = '-60.708861299999997'
23 dfTrip.at[2, 'Start_X'] = '-32.9170077'
24 dfTrip.at[2, 'Start_Address'] = 'Campbell 999'
25 dfTrip.at[2, 'StartDepotName'] = '135 - Asociación Civil Centro Diplomático Rosario'
26 dfTrip.at[2, 'End_Y'] = '-60.70604500000000'
27 dfTrip.at[2, 'End_X'] = '-32.93193'
28 dfTrip.at[2, 'EndDepotName'] = '119 - Cooperativa el progreso'
29 dfTrip.at[2, 'Name'] = 'Route 3'
30
31
32 dfTrip['TravelTime'], dfTrip['Distance'] = None, None
33 dfTrip
```

Cargamos las rutas desde OpenStreetMaps. Primero indicamos los stops que va a tener. Luego se carga el resultado de la consulta, en la que traigo para cada sub ruta la distancia y el tiempo total de viaje (en minutos).

```
In [674]: 1 route_layer = network.RouteLayer(route_service_url, gis=gis)
2
3 for idx, values in dfTrip.iterrows():
4     stops = '{0},{1}; {2},{3}'.format(values['Start_Y'], values['Start_X'],
5                                       values['End_Y'], values['End_X'])
6     #print(stops)
7     result = route_layer.solve(stops=stops, directions_language='es-AR', return_routes=True,
8                               return_stops=True, return_directions=True,
9                               output_lines='esriAOOutputLineNone',
10                                return_barriers=False, return_polygon_barriers=False,
11                                return_polyline_barriers=False)['routes'][0]['attributes']
12     dfTrip.loc[idx, 'TravelTime'] = result['Total_TravelTime']
13     dfTrip.loc[idx, 'Distance'] = result['Total_Kilometers']
14
15
16 #muestro la tabla con las tres rutas y las estimaciones, es loc1->loc2;loc2->loc3;loc3->loc4
17 dfTrip
```

```
Out[674]:
```

	Start_X	Start_Y	Start_Address	StartDepotName	End_X	End_Y	EndDepotName	Name	TravelTime	Distance
0	-32.9932693	-60.68061139999999	Dorrego 6178	56 - Cielos Abiertos	-32.9937538	-60.63090030000001	390 - Parroquia Nuestra Señora del Valle (Centro de Vida)	Route 1	7.33818	3.27552
1	-32.9937538	-60.63090030000001	Cucha Cucha 4820	390 - Parroquia Nuestra Señora del Valle (Centro de Vida)	-32.9170077	-60.70886129999997	135 - Asociación Civil Centro Diplomático Rosario	Route 2	22.5583	20.2295
2	-32.9170077	-60.70886129999997	Campbell 999	135 - Asociación Civil Centro Diplomático Rosario	-32.93193	-60.7080450000000	119 - Cooperativa el progreso	Route 3	5.89787	2.30277

Cargamos la ruta para los puntos marcados.

Un punto relevante a marcar aquí. Si bien la capa utilizada de Open Street Maps, route_layer, y el método utilizado -solve- son abiertos para los desarrolladores en general, el método solve tiene una serie de parámetros que no pueden ser utilizados libremente: los parámetros preserve_first_stop, find_best_sequence y preserve_last_stop son parámetros de optimización de rutas que están reservados.

En los casos de tener varios nodos pueden reorganizar las rutas para optimizarlas, son casos que se recomendarían fuertemente de implementar, pero que aquí no lo haremos. Dejamos los parámetros comentados para recordar cuál hubiera sido la implementación recomendada.

```
In [675]: 1 #cargó un tiempo de inicio y coordenadas de prueba, cargo la libreria datetime
2 startTime = int(datetime.datetime.utcnow().timestamp() * 1000)
3
4 stopsTrip='''-60.66061139999999,-32.9932693;-60.63090030000001,-32.9937538;-60.70886129999997,-32.9170077'''
5
6 #dejamos comentados los dos parametros de la capa superior -paga- que permitirian reorganizar el viaje
7 #Los parámetros son preserve_first y find_best_sequence
8
9 resultTrip = route_layer.solve(stops=stopsTrip,
10                               directions_language='es-AR',
11                               #preserve_first_stop=True,
12                               #find_best_sequence=True,
13                               return_directions=True, return_routes=True,
14                               return_stops=True,
15                               return_barriers=False, return_polygon_barriers=False,
16                               return_polyline_barriers=False,
17                               start_time=startTime,
18                               start_time_is_utc=True)
19
20 travel_time = resultTrip['routes'][0]['features'][0]['attributes']['total_travel_time']
21 print("El tiempo total de viaje es de {0:.2f} min".format(travel_time))
22 #print(resultTrip['routes'][0]['features'][0]['geometry']['paths'])
23
El tiempo total de viaje es de 28.67 min
```

Vemos el recorrido

```
In [676]: 1 recordsTrip = []
2
3 #genero e inicializo las variables donde voy a cargar los indicadores en cada iteración
4
5 travel_time, time_counter = 0, 0
6 distance, distance_counter = 0, 0
7
8 #recorro el resultado obtenido cargo en el objeto record
9 for i in resultTrip['directions'][0]['features']:
10     time_of_day = datetime.datetime.fromtimestamp(i['attributes']['arriveTimeUTC'] / 1000).strftime('%H:%M:%S')
11     time_counter = i['attributes']['time']
12     distance_counter = i['attributes']['length']
13     travel_time += time_counter
14     distance += distance_counter
15     recordsTrip.append( (time_of_day, i['attributes']['text'],
16                        | round(travel_time, 2), round(distance, 2)) )
17
In [677]: 1 from IPython.display import HTML
2
3 #cargó el panda
4 pd.set_option('display.max_colwidth', 100)
5 df = pd.DataFrame.from_records(recordsTrip, index=[i for i in range(1, len(recordsTrip) + 1)],
6                               columns=['Hora', 'Indicacion',
7                                       'Duracion (min)', 'Distancia (km)'])
8 HTML(df.to_html(index=False))
```

OUT[677]:	Hora	Indicacion	Duracion (min)	Distancia (km)
	19:03:57	Salga desde Location 1	0.00	0.00
	19:03:57	Diríjase al este por Bahía Juárez hacia Melán	2.61	0.66
	19:06:34	En la señal de stop, gire a la derecha en Sarmiento	2.95	0.76
	19:06:54	Gire a la izquierda por Salamanca	3.39	0.86
	19:07:20	Gire a la izquierda por Avenida San Martín y después gire a la derecha por J. María Gutiérrez	3.80	0.96
	19:07:46	Siga adelante por Holmberg	4.02	1.07
	19:07:56	Gire a la izquierda por Buenos Aires	4.23	1.14
	19:08:11	Gire a la derecha por J. M. Gutiérrez	5.29	1.56
	19:09:15	Gire a la derecha por J. María Gutiérrez	5.38	1.62
	19:09:20	Siga adelante por Lamadrid	5.44	1.64
	19:09:24	Gire a la izquierda por Esteban de Luca	6.78	1.99
	19:10:43	Gire a la izquierda por Cucha Cucha	6.96	2.04
	19:10:56	Ha llegado a Location 2, que se encuentra a la izquierda	6.96	2.04
	19:10:56	Salga de Location 2	6.96	2.04
	19:10:56	Siga hacia el norte por Cucha Cucha	7.06	2.06
	19:11:02	Gire a la izquierda por J. M. Gutiérrez	8.47	2.40
	19:12:25	Gire a la izquierda por J. María Gutiérrez	8.56	2.44
	19:12:30	Siga adelante por Lamadrid	9.18	2.65
	19:13:06	En el semáforo, siga adelante por Regimiento 11	9.69	2.80
	19:13:39	En el semáforo, siga adelante por Ntra. Sra. Rosario	9.96	2.88
	19:13:55	En el semáforo, siga adelante por Ayacucho	10.86	3.11
	19:14:49	En el semáforo, siga adelante por Láinez	11.10	3.18
	19:15:03	Siga adelante por Edison	11.35	3.25
	19:15:18	En el semáforo, siga adelante por Ruf. Barbosa	11.47	3.29
	19:15:25	Siga adelante por Cabildo	11.72	3.36
	19:15:40	Siga adelante por Muñoz	12.01	3.43
	19:15:56	En el semáforo, siga adelante por Ayacucho	12.53	3.62
	19:16:29	Siga adelante por Calle 529	12.63	3.65
	19:16:36	Gire a la derecha por Ayacucho	13.04	3.83
	19:17:00	Siga adelante por Juan B. Justo	13.18	3.88
	19:17:06	Gire a la derecha por la rampa hacia Va. Gdor. Gálvez / Buenos Aires / Venado Tuerto	24.56	12.97
	19:28:31	Siga adelante por Avenida de Circunvalación 25 de Mayo Bis	24.96	13.32
	19:28:55	Tome la rampa de la derecha por Av. J. J. Paso	25.30	13.48
	19:29:15	Gire a la izquierda por Juan Pablo II Bis	25.46	13.51
	19:29:25	En el semáforo, gire a la derecha en Juan José Paso	26.53	14.04
	19:30:35	En la rotonda, tome la tercera salida por Avenida Provincias Unidas Bis	27.49	14.32
	19:31:26	Gire a la derecha por José Ingenieros	28.54	14.60
	19:32:36	Gire a la izquierda por Campbell Bis	28.67	14.61
	19:32:37	Ha llegado a Location 3, que se encuentra a la izquierda	28.67	14.61

Generamos la visualización de la ruta.

```

In [678]: 1 #route_layer = network.RouteLayer(route_service_url, gis=gis)
2
3 resultTripMap = route_layer.solve(stops=stopsTrip,
4                                   directions_language='es-AR', return_routes=True,
5                                   return_stops=True, return_directions=False,
6                                   return_barriers=False, return_polygon_barriers=False,
7                                   return_polyline_barriers=False)
8
9 my_map2 = gis.map('Rosario, Santa Fe', zoomlevel=12)
10
11 my_map2.clear_graphics()
12 stop_count = resultTripMap['routes'][0]['features'][0]['attributes']['StopCount']
13 travel_time = resultTripMap['routes'][0]['features'][0]['attributes']['Total_TravelTime']
14 distance = resultTripMap['routes'][0]['features'][0]['attributes']['Total_Kilometers']
15
16 data = [['Number of stops', stop_count],
17         ['Total travel time', '{0:.2f} min'.format(travel_time)],
18         ['Total travel distance', '{0:.2f} km'.format(distance)]]
19
20 df = pd.DataFrame.from_records(data)
21 styles = [
22     dict(selector="td", props=[("padding", "2px")]),
23     dict(selector=".row_heading, .blank", props=[('display', 'none;')]),
24     dict(selector=".col_heading, .blank", props=[('display', 'none;')])
25 ]
26
27 symbol = {
28     "type": "esriSLS",
29     "style": "esriSLSolid",
30     "color": [128,0,128,90],
31     "width": 4
32 }
33
34 popup_route = {"title": "Route",
35                "content": df.style.set_table_styles(styles).render()}
36 popup_stop = {"title": "Stop {} ",
37               "content": df.style.set_table_styles(styles).render()}
38
39 my_map2.draw(resultTripMap['routes'][0]['features'][0]['geometry'], popup_route, symbol)
40
41 for stop in resultTripMap['stops']['features']:
42     address = geocoding.reverse_geocode(stop['geometry'])['address']['Match_addr']
43     my_map2.draw(stop['geometry'],
44                  {"title": "Stop {}".format(stop['attributes']['Sequence']),
45                   "content": address})
45
46 my_map2.zoom = 12
47 my_map2

```



- **Buscar si existe ruta**

En este caso busco un recorrido específico que de antemano sé que no existe. Me garantizo que no exista ninguna relación previa entre dos nodos

Por el caso de uso ellos entregarán el orden de los nodos en estos casos, pero se puede recomendar utilizar algoritmos de reordenamiento a mediano plazo, como los métodos ya

nombrados en la documentación de las capas `route_analysis` y `network_analysis` de OpenStreetMap.

```
In [602]: 1 #paso una secuencia de id que sé que no existe
2 addresslist="'33cee5e8-3ac3-4ca8-8cb0-269f54d96fb2','1a509e68-0b2a-470b-a112-d8ffa4fa0d6f','a9135abe-20e7-4932-9f67-903ccadb94c','59233fd5-db4b-4990-b95f-79c18db1af34'"
3
4 #genero una lista que utilizare para dar de alta las relaciones en caso de no existir
5 list_address=addresslist.split(',')

In [479]: 1
2 #verifico que no haya relaciones entre ningún nodo de los existentes
3 query_neo="MATCH (n:Dropoff) where n.idAddress IN [" + addresslist + "] WITH collect(n) as nodes UNWIND nodes as n UNWIND nodes as a
4
5 print(query_neo)
6
7 with driver.session() as session:
8
9     nodes_route= session.run(query_neo)
10     result_nodes = [record for record in nodes_route.data()]
11     #si no hay relacion, las construyo
12     if len(result_nodes)==0:
13         print("no hay relaciones, las creo")
14         #creo las relaciones del modo A->B->C->D
15         idx=0
16         while idx < (len(list_address)-1):
17             print("Ruta " + str(idx))
18             create_query="MATCH (a:Dropoff), (b:Dropoff) WHERE (a.idAddress='"+list_address[idx]+"') and (b.idAddress='"+list_add
19             print(create_query)
20             session.run(create_query)
21             idx+=1
22
23     else:
24         print("Existe la ruta, la recupero")
25         i=0
26         stop_points=""
27         while i < (len(list_address)):
28             exist_query="MATCH (d:Dropoff) WHERE (d.idAddress='"+list_address[i]+"') RETURN d"
29             result=session.run(exist_query)
30             node=[record for record in result.data()]
31             long_stop=str(node[0]['d']['longitude'])
32             lat_stop=str(node[0]['d']['latitude'])
33             stop_points+=long_stop+', '+lat_stop+';'
34             i+=1
35             print(stop_points)
36
37 driver.close()
```

```
MATCH (n:Dropoff) where n.idAddress IN ['33cee5e8-3ac3-4ca8-8cb0-269f54d96fb2','1a509e68-0b2a-470b-a112-d8ffa4fa0d6f','a9135abe-20e7-4932-9f67-903ccadb94c','59233fd5-db4b-4990-b95f-79c18db1af34'] WITH collect(n) as nodes UNWIND nodes as n UNWIND nodes as m WITH * WHERE id(n) < id(m) MATCH path = allShortestPaths((n)-[*..4]-(m)) RETURN path
Existe la ruta, la recupero
-60.62530019999997,-32.9744581;
-60.62530019999997,-32.9744581;-60.73866570000001,-32.9341162;
-60.62530019999997,-32.9744581;-60.73866570000001,-32.9341162;-60.70250020000003,-32.9258416;
-60.62530019999997,-32.9744581;-60.73866570000001,-32.9341162;-60.70250020000003,-32.9258416;-60.734327,-32.847153;
```

Traemos la ruta y, como en otros casos, la resolvemos vía Open Street Maps. Mostramos solo el tiempo general.

```
In [504]: 1 #cargo un tiempo de inicio y coordenadas de prueba, cargo la libreria datetime
2 startTime = int(datetime.datetime.utcnow().timestamp() * 1000)
3
4 resultTrip2 = route_layer.solve(stops=stop_points,
5                                 directions_language='es-AR',
6                                 #preserve_first_stop=True,
7                                 #find_best_sequence=True,
8                                 return_directions=True, return_routes=True,
9                                 return_stops=True,
10                                return_barriers=False, return_polygon_barriers=False,
11                                return_polyline_barriers=False,
12                                start_time=startTime,
13                                start_time_is_utc=True)
14
15 travel_time = resultTrip2['routes'][0]['features'][0]['attributes']['Total_TravelTime']
16 print("El tiempo total de viaje es de {0:.2f} min".format(travel_time))
17 #print(result['routes'][0]['features'][0]['geometry']['paths'])
18
El tiempo total de viaje es de 44.33 min
```

• Alta de Incidentes

En juego de datos de prueba generamos una ruta con incidentes marcados.

Una parte de lo requerido es, al recuperar una ruta que se advierta si la misma ha tenido últimamente algún incidente. Como marcamos en el documento, no tenemos un criterio específico o tipología para marcar los incidentes, por lo cual lo que haremos aquí es registrar un par de campos (una descripción y una fecha), así como aumentar en 5 el peso del tramo en cuestión.

Está claro que esta parte es modificable, pero en lo esencial se trata de registrar los datos y aumentar un peso.

```
In [662]: 1 #recupero las propiedades del camino
2 retrieve_query="MATCH (from:Dropoff {idAddress:'373358dc-69cc-498b-abad-8532d89e53af'})-[r:ROAD]->(to:Dropoff {idAddress:'45cc
3
4 with driver.session() as session:
5
6     nodes=session.run(retrieve_query)
7     result_nodes = [record for record in nodes.data()]
8
9 #imprimo las propiedades de la ruta
10 print(result_nodes[0])
11
12 driver.close()
13
```

```
{'PROPERTIES(r)': {'date_events_reported': ['2021-03-16', '2021-03-16', '2021-03-11', '2021-03-07', '2021-03-09'], 'wgt': 50,
'event_description': ['Piquete en olla popular en calle X', 'Piquete en olla popular en calle X', 'Disturbios con la policia',
'Piquete en la zona', 'Calle en refaccion']}}
```

A continuación se muestra el peso de la ruta, así como los incidentes que se han registrado en la misma últimamente. Estos datos son manejables cada vez que recuperamos una ruta.

```
In [664]: 1 print("Peso de la Ruta: "+ str(result_nodes[0]['PROPERTIES(r)']['wgt']))
2
3 idx=0
4 count_events=len(result_nodes[0]['PROPERTIES(r)']['event_description'])
5 while idx < len(result_nodes[0]['PROPERTIES(r)']['event_description']):
6     print("Incidentes Reportados: "+str(result_nodes[0]['PROPERTIES(r)']['event_description'][idx])+" - Fecha: "+str(result_
7         idx+=1
8
```

```
Peso de la Ruta: 50
Incidentes Reportados: Piquete en olla popular en calle X - Fecha: 2021-03-16
Incidentes Reportados: Piquete en olla popular en calle X - Fecha: 2021-03-16
Incidentes Reportados: Disturbios con la policia - Fecha: 2021-03-11
Incidentes Reportados: Piquete en la zona - Fecha: 2021-03-07
Incidentes Reportados: Calle en refaccion - Fecha: 2021-03-09
```

Supongamos ahora que, ante un nuevo incidente a reportar agregamos 5 puntos al peso de la ruta. El mismo deberá venir con los datos específicos de lo acontecido.

Los valores específicos y hasta una tipología de incidentes son manejables, lo importante es dar cuenta de cómo impactar los datos de los incidentes en esta fase. Por esa razón es que comentamos `event_type` y `weight`, los cuales pueden ser utilizados sin mayor problema.

En este caso estamos agregando la descripción y fecha del evento, y agregándole 5 al peso que ya tenía la ruta, lo cual obviamente puede modificarse por otro criterio.

```
In [665]: 1 #ingreso los datos
2 description='Maquinaria vial reparando la calle'
3 date_event = '2021-03-18'
4 #event_type = 2
5 #weight = event_type * value
6
7 with driver.session() as session:
8     basic_query = "MATCH (from:Dropoff {idAddress:'373358dc-69cc-498b-abad-8532d89e53af'})-[r:ROAD]->(to:Dropoff {idAddress:'4"
9     set_weight="SET r.wgt=r.wgt+5 "
10    set_description = "SET r.event_description=['"+description+"']+r.event_description "
11    set_data="SET r.date_events_reported=['"+date_event+"']+r.date_events_reported "
12    set_return="RETURN PROPERTIES(r)"
13    update_event=basic_query + set_weight + set_description + set_data + set_return
14    print(update_event)
15
16    relation_update=session.run(update_event)
17    result_relation = [record for record in relation_update.data()]
18
19 driver.close()
```

MATCH (from:Dropoff {idAddress:'373358dc-69cc-498b-abad-8532d89e53af'})-[r:ROAD]->(to:Dropoff {idAddress:'45cc339e-a86b-4784-9976-f3ade88eb2d8'}) SET r.wgt=r.wgt+5 SET r.event_description=['Maquinaria vial reparando la calle']+r.event_description SET r.date_events_reported=['2021-03-18']+r.date_events_reported RETURN PROPERTIES(r)

```
In [667]: 1 print("Peso de la Ruta: "+ str(result_relation[0]['PROPERTIES(r)']['wgt']))
2
3 idx=0
4 count_events=len(result_relation[0]['PROPERTIES(r)']['event_description'])
5 while idx < count_events:
6     print(str(idx)+" - Incidentes Reportados: "+str(result_relation[0]['PROPERTIES(r)']['event_description'][idx])+" - Fecha
7     idx+=1
8
9
Peso de la Ruta: 55
0 - Incidentes Reportados: Maquinaria vial reparando la calle - Fecha: 2021-03-18
1 - Incidentes Reportados: Piquete en olla popular en calle X - Fecha: 2021-03-16
2 - Incidentes Reportados: Piquete en olla popular en calle X - Fecha: 2021-03-16
3 - Incidentes Reportados: Disturbios con la policia - Fecha: 2021-03-11
4 - Incidentes Reportados: Piquete en la zona - Fecha: 2021-03-07
5 - Incidentes Reportados: Calle en refaccion - Fecha: 2021-03-09
```

13 ANEXO IV - Análisis detallado de datos

Tabla “addresses” exploración - Vista inicial

1 SELECT *FROM `nilus-bi.nilus_logistics.addresses` LIMIT 1000														
Ubicación de procesamiento: US														
Ejecutar Guardar consulta Guardar vista Programar consulta Más														
Esta consulta procesará 2.2 MB cuando se ejecute														
Resultados de la consulta														
Se ha completado la consulta (tiempo transcurrido: 1.2 s; bytes procesados: 2.2 MB)														
Información de la tarea														
ResultadosJSONDetalles de ejecución														
Fila	zip_code	street_1	contact_email	country	city	gateway_data	id	lookup	_sdc_table_version	updated_at	_sdc_received_at	institution_id	_sdc_sequence	
1	null	Bv. 27 de Febrero 3699	null	AR	Rosario	0	9fa1928c-6c14-4166-ad34-15767ab22437	null	1565122950606	2018-06-25 20:26:14.121 UTC	2019-08-06 20:22:30.993 UTC	71f0f683-bffa-4189-a503-88aa5322e399	1565122950606	
2	null	Bv. 27 de Febrero 3699	null	AR	Rosario	0	70f675a9-3d8d-4ed8-8aed-b8640d1cd50c	null	1565122950606	2018-06-25 20:26:14.872 UTC	2019-08-06 20:22:30.993 UTC	1f2968bb-08de-422a-a612-fb2a3585de1b	1565122950606	
3	null	Mariano Moreno 61	colo@nilus.org	AR	Rosario	0	d395623a-ef2d-4d9f-bab9-a549d92ee57b	null	1565122950606	2018-07-13 00:47:14.643 UTC	2019-08-06 20:22:30.993 UTC	40e523ed-3ac4-48a9-a76d-bd93b7a44d5	1565122950606	
4	null	La República 6649		AR	Rosario	0	dbf3d446-ee2-43b2-8ca7-a963dc562676	null	1565122950606	2018-07-13 00:48:15.836 UTC	2019-08-06 20:22:30.993 UTC	dc902ech-17ca-4d70-a5bd-d209141df8	1565122950606	
5	null	Campbell 999	null	AR	Rosario	0	927c7707-3a9f-4d32-ae8b-146c2c74123f	null	1565122950606	2018-07-13 02:19:27.646 UTC	2019-08-06 20:22:30.993 UTC	8063c9df-23e4-4e38-97d1-9f5b708f61ba	1565122950606	
6	null	Vélez Sársfield 5763	null	AR	Rosario	0	e7175851-2f5f-41a7-a961-c339b3a5d7f3	null	1565122950606	2018-07-13 02:24:32.054 UTC	2019-08-06 20:22:30.993 UTC	9ae7f5d6-eteef-4e67-adb1-2fbfd4c10272	1565122950606	
7	null	Humberto Primo 1255	null	AR	Rosario	0	8118ee0b-0c80-49c2-8489-c4363799ed0b	null	1565122950606	2018-07-25 14:16:23.227 UTC	2019-08-06 20:22:30.993 UTC	935897ae-2dc3-489a-88e9-231edf9a2bf9	1565122950606	
8	null	Humberto Primo 1255	null	AR	Rosario	0	d6060499-8c0e-4b21-a8b8-cacd17994ed7	null	1565122950606	2018-07-25 14:16:23.271 UTC	2019-08-06 20:22:30.993 UTC	564c892e-952e-4c41-8999-cdc8aeb653e9	1565122950606	
9	null	Vélez Sársfield 5763	joseleines65@yahoo.com.ar	AR	Rosario	0	a9135abe-20e7-4932-9f67-903cceed894c	null	1565122950606	2018-07-25 15:24:32.491 UTC	2019-08-06 20:22:30.993 UTC	a204c725-1f2b-42ca-8a4a-2c49e979923a	1565122950606	
10	null	Junín 1836	null	AR	Rosario	0	5a2b543b-5800-4c23-a876-94e735c108c2	null	1565122950606	2018-07-25 15:57:29.199 UTC	2019-08-06 20:22:30.993 UTC	505f6421-7914-4623-aed9-06056863d79b	1565122950606	
11	null	Pasco 3737	null	AR	Rosario	0	ee87d961-e650-4d7b-9fde-ee04bdcde093	null	1565122950606	2018-07-31 16:21:42 UTC	2019-08-06 20:22:30.993 UTC	2a3996db-f1cc-41af-9b62-f2c095c4b374	1565122950606	
12	null	Hutchinson 4241	null	AR	Rosario	0	9af2070e-22ee-4caa-bc9c-a81cd7b1c2ca	null	1565122950606	2018-08-15 12:41:20.370 UTC	2019-08-06 20:22:30.993 UTC	470ee2df-c0fb-4118-ae9b-b84803ba31e3	1565122950606	
13	null	Rio de Janeiro 3437	null	AR	Rosario	0	373358dc-69cc-498b-8bad-8532b89e53af	null	1565122950606	2018-08-16 02:49:26.456 UTC	2019-08-06 20:22:30.993 UTC	af3905c9-287e-4994-9971-dfeae67a7473	1565122950606	
14	null	Ecuador 2069	null	AR	Rosario	0	fc7880f5-17a6-42e2-88ee-2321e4d1c615	null	1565122950606	2018-08-16 02:50:33.804 UTC	2019-08-06 20:22:30.993 UTC	7e0d18a8-4ca5-4dd4-b8d5-308ca4cea12e	1565122950606	
15	null	Riobamba 739	keserfaty@gmail.com	AR	Rosario	0	89f163a7-3ec2-48ec-a532-23f1e4d1c615	null	1565122950606	2018-08-17 13:08:50.808 UTC	2019-08-06 20:22:30.993 UTC	null	1565122950606	
16	null	Av. Provincias Unidas 2050	null	AR	Rosario	0	130586c8-f371-401a-887a-7f59feaba286	null	1565122950606	2018-08-21 17:22:42.495 UTC	2019-08-06 20:22:30.993 UTC	b9dc3009-9365-42f1-b44f-60d49267a18c	1565122950606	
17	null	Pje. 1112 3317	null	AR	Rosario	0	15d5bdf3-7543-4a4b-b714-c3b2b25ac05f	null	1565122950606	2018-08-21 17:43:25.606 UTC	2019-08-06 20:22:30.993 UTC	1b9a205b-d5d3-4ab5-9e4b-64559af0e694	1565122950606	
18	null	Maradona Bis 958	null	AR	Rosario	0	913d968f-cff5-43f1-856b-c3e5409a3abd	null	1565122950606	2018-08-22 17:32:19.544 UTC	2019-08-06 20:22:30.993 UTC	70a28373-ae0e-484b-9f42-b2493118b78c	1565122950606	
19	null	Sarratea 1250	null	AR	Rosario	0	72f5c04b-c750-4d44-9a45-0a82b67df714	null	1565122950606	2018-08-22 17:34:09.103 UTC	2019-08-06 20:22:30.993 UTC	1bb0993d-ba98-40a6-80fe-63b5890606b4	1565122950606	
20	null	Juan B. Justo 7829	null	AR	Rosario	0	fa58fc02-3ced-4ed7-ae0b-73dc94847d70	null	1565122950606	2018-08-22 17:37:35.857 UTC	2019-08-06 20:22:30.993 UTC	12df325c-9f31-4645-bd5a-d54a346079de	1565122950606	
21	null	San Juan 764	null	AR	Rosario	0	f763eeaf-02d5-4eda-a902-c8f74682be8	null	1565122950606	2018-08-22 17:40:35.511 UTC	2019-08-06 20:22:30.993 UTC	6c3f31e1-9d77-48aa-b5bb-2a85033ae41d	1565122950606	
22	null	Av. Rivarola 7753	null	AR	Rosario	0	3a98b3a7-9254-400b-bba3-898d278999b9	null	1565122950606	2018-08-22 18:33:52.050 UTC	2019-08-06 20:22:30.993 UTC	9807f150-8927-4768-b092-8bf8b33ae61d	1565122950606	
23	null	Uguita 9383	null	AR	Rosario	0	1a5046b8-0b2a-470b-a112-d8ffaf4a0d6f	null	1565122950606	2018-08-22 19:00:39.555 UTC	2019-08-06 20:22:30.993 UTC	77a2a33f-b271-418c-89a6-508925b93d4d	1565122950606	
24	null	Cabal 1252	null	AR	Rosario	0	913843d0-3444-4a9a-a923-297eb6717f07	null	1565122950606	2018-08-22 19:13:45.798 UTC	2019-08-06 20:22:30.993 UTC	e7f25756-8306-4acc-8717-04e729887b32	1565122950606	
25	null	Chacabuco 4058	null	AR	Rosario	0	c4e322f2-bc10-4566-9edf-e4294d99606	null	1565122950606	2018-08-22 19:16:54.706 UTC	2019-08-06 20:22:30.993 UTC	6b66f791-c575-4366-b6bb-02646bb4c8bb	1565122950606	

Vista inicial parcial de campos relevantes.

SELECT street_1, city, country, state, updated_at, notes FROM `nilus-bi.nilus\_logistics.addresses` LIMIT 20

Editor de consultas						
1 SELECT street_1, city, country, state, updated_at, notes FROM `nilus-bi.nilus_logistics.addresses` LIMIT 20						
Ejecutar Guardar consulta Guardar vista Programar consulta Más						
Resultados de la consulta						
Se ha completado la consulta (tiempo transcurrido: 0.2 s; bytes procesados: 577.6 kB)						
Información de la tarea						
ResultadosJSONDetalles de ejecución						
Fila	street_1	city	country	state	updated_at	notes
1	Bv. 27 de Febrero 3699	Rosario	AR	Santa Fe	2018-06-25 20:26:14.121 UTC	Es el mercado en el cual el BAR tiene un espacio de clasificación de frutas y verduras.
2	Bv. 27 de Febrero 3699	Rosario	AR	Santa Fe	2018-06-25 20:26:14.872 UTC	Es el mercado en el cual el BAR tiene un espacio de clasificación de frutas y verduras.
3	Mariano Moreno 61	Rosario	AR	Santa Fe	2018-07-13 00:47:14.643 UTC	null
4	La República 6649	Rosario	AR	Santa Fe	2018-07-13 00:48:15.836 UTC	null
5	Campbell 999	Rosario	AR	Santa Fe	2018-07-13 02:19:27.646 UTC	null
6	Vélez Sársfield 5763	Rosario	AR	Santa Fe	2018-07-13 02:24:32.054 UTC	null
7	Humberto Primo 1255	Rosario	AR	Santa Fe	2018-07-25 14:16:23.227 UTC	null
8	Humberto Primo 1255	Rosario	AR	Santa Fe	2018-07-25 14:16:23.271 UTC	null
9	Vélez Sársfield 5763	Rosario	AR	Santa Fe	2018-07-25 15:24:32.491 UTC	Entrega Copa de leche
10	Junín 1836	Rosario	AR	Santa Fe	2018-07-25 15:57:29.199 UTC	null
11	Pasco 3737	Rosario	AR	Santa Fe	2018-07-31 16:21:42 UTC	null
12	Hutchinson 4241	Rosario	AR	Santa Fe	2018-08-15 12:41:20.370 UTC	null
13	Rio de Janeiro 3437	Rosario	AR	Santa Fe	2018-08-16 02:49:26.456 UTC	null
14	Ecuador 2069	Rosario	AR	Santa Fe	2018-08-16 02:50:33.804 UTC	null
15	Riobamba 739	Rosario	AR	Santa Fe	2018-08-17 13:08:50.808 UTC	Algunas notas
16	Av. Provincias Unidas 2050	Rosario	AR	Santa Fe	2018-08-21 17:22:42.495 UTC	null
17	Pje. 1112 3317	Rosario	AR	Santa Fe	2018-08-21 17:43:25.606 UTC	null
18	Maradona Bis 958	Rosario	AR	Santa Fe	2018-08-22 17:32:19.544 UTC	null
19	Sarratea 1250	Rosario	AR	Santa Fe	2018-08-22 17:34:09.103 UTC	null
20	Juan B. Justo 7829	Rosario	AR	Santa Fe	2018-08-22 17:37:35.857 UTC	null

Cantidad de datos: ¿Total de direcciones registradas?

1	SELECT count(*) FROM `nilus-bi.nilus_logistics.addresses`
---	---

Ubicación de procesamiento: US

Ejecutar

Guardar consulta

Guardar vista

Programar consulta

Más

Resultados de la consulta

GUARDAR RESULTADOS

EXAMINAR LOS DATOS

Se ha completado la consulta (tiempo transcurrido: 0,6 s; bytes procesados: 0 B)

Información de la tarea

Resultados

JSON

Detalles de ejecución

Fila	f0_
1	11386

Calidad y cantidad de datos: ¿Códigos postales diferentes?

Editor de consultas	
1	SELECT distinct zip_code FROM `nilus-bi.nilus_logistics.addresses`

Ejecutar

Guardar consulta

Guardar vista

Programar consi

Resultados de la consulta

GUARDAR RESULTADOS

Se ha completado la consulta (tiempo transcurrido: 2,1 s; bytes procesados: 180 B)

Información de la tarea

Resultados

JSON

Detalles de ejecución

Fila	zip_code
1	S2003
2	S2007BQI
3	S2009EBP
4	S2004INC
5	S2011KDF
6	S2008ANW
7	S2003AWW
8	S2003EQB
9	Malvinas
10	null
11	S2007
12	S2001ARQ
13	S2001GKB
14	S2007BQT
15	S2002
16	S2006
17	S2003FFA
18	Manuel

Calidad y cantidad de datos: ¿direcciones diferentes?

Editor de consultas

1

SELECT distinct street_1 FROM `nilus-bi.nilus\_logistics.addresses`

Ejecutar

Guardar consulta

Guardar vista

Programar consi

Resultados de la consulta

GUARDAR RESULTADOS

Se ha completado la consulta (tiempo transcurrido: 0,7 s; bytes procesados: 241,6 kB)

Información de la tarea

Resultados

JSON

Detalles de ejecución

45

Pje. 112 3317

46

Sarratea 1250

47

San Juan 764

48

Av. Rivarola 7753

49

Urquiza 9383

50

Chacabuco 4058

51

España 6417

52

España 7180

53

5 de Agosto 2102

54

Av. Sorrento 1400

55

Av. Pellegrini 3057

56

Humberto Primo 5330

57

Tupac Amaru 6248

58

Av. Soldado Alberto Aguirre 2982

59

Buenos Aires 6354

60

Rueda 359

61

Amenabar 4325

62

Culaciatti 5994

63

Pasaje Puerto Belgrano

64

Riobamba 5399

65

G. Carrasco 2070

66

Río Atuel 4001

67

Gorriti 6080

68

Aborígenes Argentinos 6016

69

Entre Ríos 6288

70

Felipe Moré 763

71

null

Calidad y cantidad de datos: ¿Países?

Editor de consultas

1
SELECT distinct country FROM `nilus-bi.nilus\_logistics.addresses`

Ejecutar
Guardar consulta
Guardar vista
Programar consi

Resultados de la consulta

GUARDAR RESULTADOS

Se ha completado la consulta (tiempo transcurrido: 0,6 s; bytes procesados: 46,2 kB)

Información de la tarea
Resultados
JSON
Detalles de ejecución

Fila	country
1	AR
2	Argentina

Calidad y cantidad de datos: ¿Ciudades?

Editor de consultas

1
SELECT distinct city FROM `nilus-bi.nilus\_logistics.addresses`

Ejecutar
Guardar consulta
Guardar vista
Programar consi

Resultados de la consulta

GUARDAR RESULTADOS

Se ha completado la consulta (tiempo transcurrido: 0,5 s; bytes procesados: 103,7 kB)

Información de la tarea
Resultados
JSON
Detalles de ejecución

Fila	city
1	Artigas 5455
2	Rosario
3	Argentinas 456

Calidad y cantidad de datos: ¿id's de addresses?

1 SELECT distinct id FROM `nilus-bi.nilus\_logistics.addresses`

Ejecutar

Guardar consulta

Guardar vista

Programar consulta

Más

Resultados de la consulta

GUARDAR RESULTADOS

EXAMINAR LOS DATOS

Se ha completado la consulta (tiempo transcurrido: 0,8 s; bytes procesados: 438,6 kB)

Información de la tarea

Resultados

JSON

Detalles de ejecución

56	3a9603a7-9204-4000-80a3-696027909909	
57	1a509e68-0b2a-470b-a112-d8ffa4fa0d6f	
58	91384340-3444-4a9a-a92f-297eb8717f07	
59	bf261b56-def0-4920-aaa1-9744c49530dc	
60	9cf33d1e-ffb3-4ca6-a1a8-60f2e6961340	
61	db4d047e-b82e-4704-bd36-eedb5e9ec4bb	
62	aec4f435-1496-4688-81d8-feed00576556	
63	6a23d494-7bd8-412b-9dfb-f0f5c9a680af	
64	53457c6d-4266-4b3c-b820-0138f788c711	
65	de160c1e-f8fa-4490-b0bf-4df554d0f622	
66	0c20258b-ea8e-4092-9f07-5c6138d252b9	
67	fe8c5dc8-6234-4a28-81ec-eff67c620dcc	
68	5d01cf1a-e33d-43cc-a8eb-755b520b3495	
69	33cee5e8-3ac3-4ca8-8cb0-269f54d96fb2	
70	61ee89aa-86a9-421d-9e3e-10c3912eb240	
71	fcf37b37-3c5a-4d9d-810f-cecae751e760	
72	864c45b4-5140-4b41-bb64-69644579031f	
73	fb29f78b-9049-4efb-ba0f-ffd0622ee471	
74	35e3f257-2aca-4e6e-9f97-e60b9fed1a14	
75	15e60758-bd74-475f-b76c-3ea88f96027c	
76	db60ea3b-a880-4799-876b-a88dce0cc585	
77	8b0c6593-da88-4038-b6cb-687154577f5b	
78	388093be-a019-4658-a4d5-99b90039fd90	
79	271e4e3e-f814-48ef-806b-2bde77477948	
80	1cfbf1e1-88e7-4b74-b574-149b38c047b0	
81	0626acc3-6067-4881-9ca5-72beac3a544b	
82	ed38b9d3-e91b-4ddc-9d66-9ad92e996293	

Hay 82 “id’s” diferentes para 70 “streets_1”?Revisar!!

Calidad y cantidad de datos: “updated_at”

1

SELECT distinct updated_at FROM `nilus-bi.nilus\_logistics.addresses`

Ejecutar

Guardar consulta

Guardar vista

Programar consulta

Más

Resultados de la consulta

GUARDAR RESULTADOS

EXAMINAR LOS DATOS

Se ha completado la consulta (tiempo transcurrido: 0,7 s; bytes procesados: 92,3 kB)

Información de la tarea

Resultados

JSON

Detalles de ejecución

56	2018-08-24 14:23:42.332 UTC	
57	2018-08-24 16:35:33.843 UTC	
58	2018-08-24 16:55:24.791 UTC	
59	2018-08-30 14:00:39.539 UTC	
60	2018-08-30 14:02:43.350 UTC	
61	2018-08-31 11:29:58.051 UTC	
62	2018-08-31 11:35:52.554 UTC	
63	2018-08-31 11:40:36.758 UTC	
64	2018-08-31 11:49:24.628 UTC	
65	2018-08-31 14:53:53.536 UTC	
66	2018-08-31 14:57:34.555 UTC	
67	2018-08-31 14:59:11.873 UTC	
68	2018-05-11 16:15:35.578 UTC	
69	2018-08-16 02:51:33.438 UTC	
70	2018-02-23 16:26:38.055 UTC	
71	2018-02-23 16:26:38.112 UTC	
72	2018-02-23 16:28:35.830 UTC	
73	2018-02-23 16:28:37.630 UTC	
74	2018-02-23 16:28:37.683 UTC	
75	2018-02-23 16:28:37.709 UTC	
76	2018-02-23 16:28:37.735 UTC	
77	2018-02-23 16:28:37.762 UTC	
78	2018-02-23 16:28:37.813 UTC	
79	2018-02-23 16:28:43.105 UTC	
80	2018-02-23 16:28:43.477 UTC	
81	2018-02-23 16:28:44.663 UTC	
82	2018-04-19 05:00:05.560 UTC	

Coincide con la cantidad de diferentes “id’s” de direcciones, es decir que el campo lo que representa en realidad es la fecha de alta de la última dirección diferente.

Calidad y cantidad de datos: “id, street_1, institution_id”

1 SELECT distinct id, street_1, institution_id FROM `nilus-bi.nilus\_logistics.addresses`

Ejecutar

Guardar consulta

Guardar vista

Programar consulta

Más

Resultados de la consulta

GUARDAR RESULTADOS

EXAMINAR LOS DATOS

Se ha completado la consulta (tiempo transcurrido: 0,0 s; almacenada en caché)

Información de la tarea

Resultados

JSON

Detalles de ejecución

27	5621ff36-843f-45c5-8323-1daf9f755427	Medici 4620	d4b41a5f-0f9d-40f9-bafa-c18c07c1e374	
28	fef36949-a1e3-492f-bcb4-333d7293d62a	Gorriti 5559	0c5cf483-9510-4ef8-8616-1a47531552f0	
29	864c45b4-5140-4b41-bb64-69644579031f	Río Atuel 4001	c1653ed6-af48-4b93-b286-f3756f800352	
30	dec3a0ff-0f7c-4860-844e-432189d8a552	Gorriti 6080	ba96a4d7-198f-4e0c-8d4c-b02b2c737600	
31	15e60758-bd74-475f-b76c-3ea88f96027c	Casilda 6003	null	
32	db60ea3b-a880-4799-876b-a88dce0cc585	Av. Provincias Unidas 2050	null	
33	8b0c6593-da88-4038-b6cb-687154577f5b	Aborígenes Argentinos 6016	null	
34	aaddc455-fd38-4187-93f8-f0a7aec69d48	Hutchinson 4241	null	
35	1f90a901-c1d8-4270-b16a-26545b7a9d19	Cucha Cucha 4820	null	
36	d08ec4a4-bc14-4c21-a3b0-7859ee4d9041	Carriego 360	null	
37	271e4e3e-f814-48ef-806b-2bde77477948	Amenabar 4325	null	
38	2109670e-ee00-4adf-8f8f-5fbe16abf5a1	null	null	
39	0626acc3-6067-4881-9ca5-72beac3a544b	null	null	
40	ed38b9d3-e91b-4ddc-9d66-9ad92e996293	Amenabar 4325	9362060e-3185-49d0-bbae-5b52650653be	
41	9fa1928c-6c14-4166-ad34-15767ab22437	Bv. 27 de Febrero 3699	71f0 added3-bffa-4189-a503-88aa53d22e39	
42	70f675a9-3d8d-4ed8-8aed-b8640d1cd50c	Bv. 27 de Febrero 3699	1f2968bb-08de-422a-a612-fb2a3585de1b	
43	d395623a-af2d-44f9-bab9-a454d92ee57b	Mariano Moreno 61	40e523ed-3ac4-48e9-a76d-bbd93b7a44d5	
44	dbf3b446-cee2-43b2-8ce7-a9b3cd562676	La República 6649	dc902ecb-17ca-4d70-a5bd-d2f09141dff8	
45	e7175851-2f5f-41a7-a961-c339b3a5d7f3	Vélez Sársfield 5763	9a67f5d6-e6ef-46f7-a0b1-2fbfd4c10272	
46	8118ee0b-0c80-49c2-8489-c4363759edb0	Humberto Primo 1255	935897ae-2dc3-489a-88e9-231ed6fa2bf9	
47	a9135abe-20e7-4932-9f67-903cceedb94c	Vélez Sársfield 5763	a204c725-1f2b-42ca-8a4a-2c49e979323a	
48	5a2b543b-5800-4c23-a87d-94e735c108c2	Junín 1836	505f6421-7914-4623-ae09-06056863d79b	
49	e8b7d961-e650-487b-9fde-eed04bcdeb03	Pasco 3737	3a795bd0-f1ce-41af-9b82-f2c095c4b374	
50	373358dc-69cc-498b-abad-8532d89e53af	Rio de Janeiro 3437	af3905c9-287e-49a4-9971-dfea687a74f3	
51	fc7880fd-17a6-42e2-88ee-2b814eb14678	Ecuador 2069	7e0d18a8-6ca5-4dd4-b8d5-308ca4cea12e	
52	130586c8-f371-401a-887e-f759feaba286	Av. Provincias Unidas 2050	b9dc3009-9365-42f1-b44f-60049267a18c	
53	b5dk5bf2-7542-4a48-b714-cbb2b25cc05f	Dic. 1112 3217	1b5b205b-d5d2-44bd-864b-64550a70a604	

No todas las direcciones tienen “institution_id y/o street_1”.

Calidad y cantidad de datos: “state”

1
SELECT distinct state FROM `nilus-bi.nilus\_logistics.addresses`

Ejecutar
Guardar consulta
Guardar vista
Programar con

Resultados de la consulta
GUARDAR RESULTADOS

Se ha completado la consulta (tiempo transcurrido: 0,8 s; bytes procesados: 115,4 kB)

Información de la tarea
Resultados
JSON
Detalles de ejecución

Fila	state
1	Santa Fe
2	C1406 CABA
3	C1440BRG CABA

Tener en cuenta para sustituir por “Santa Fe” lo que sea diferente post extracción.

Selección final y extracción de campos relevantes de la tabla “Addresses”

addresses

Esquema
Detalles
Vista previa

Nombre del campo	Tipo
zip_code	STRING
street_1	STRING
contact_email	STRING
country	STRING
city	STRING
gateway_data	STRING
id	STRING
lookup	STRING
_sdc_table_version	INTEGER
updated_at	TIMESTAMP
_sdc_received_at	TIMESTAMP
institution_id	STRING
_sdc_sequence	INTEGER
contact_name	STRING
state	STRING
contact_cellphone	STRING
created_at	TIMESTAMP
gateway	STRING
notes	STRING
_sdc_batched_at	TIMESTAMP
_sdc_extracted_at	TIMESTAMP
gateway_id	STRING

Descripción

Ninguna

Información de la tabla

ID de tabla	nilus-bi:nilus_logistics.addresses
Tamaño de la tabla	2,42 MB
Número de filas	12.659
Creación	6 ago 2019 17:47:13
Caducidad de la tabla	Nunca
Última modificación	21 ene 2021 9:31:35
Ubicación de los datos	US

```
SELECT id, street_1, country, state, city, institution_id, notes FROM `nilus-bi.nilus_logistics.addresses`
```

1 SELECT id, street_1, country, state, city, institution_id, notes FROM `nilus-bi.nilus_logistics.addresses`							
Ejecutar Guardar consulta Guardar vista Programar consulta Más							
Resultados de la consulta GUARDAR RESULTADOS EXAMINAR LOS DATOS							
Se ha completado la consulta (tiempo transcurrido: 0,1 s; almacenada en caché)							
Información de la tarea Resultados JSON							
Fila	id	street_1	country	state	city	institution_id	notes
1	9fa1928c-6c14-4166-ad34-15767ab22437	Bv. 27 de Febrero 3699	AR	Santa Fe	Rosario	71f0fdd3-bffa-4189-a503-88aa53d22e39	Es el mercado en el cual el BAR tiene un espacio de clasificación de frutas y verduras.
2	70f675a9-3d8d-4e08-8aed-b8640d1cd50c	Bv. 27 de Febrero 3699	AR	Santa Fe	Rosario	1f2968bb-08de-422a-a612-fb2a3585de1b	Es el mercado en el cual el BAR tiene un espacio de clasificación de frutas y verduras.
3	d395623a-af2d-44f9-bab9-a454d92ee57b	Mariano Moreno 61	AR	Santa Fe	Rosario	40e523ed-3ac4-48e9-a76d-bd93b7a44d5	null
4	dbf3b446-cee2-43b2-8ce7-a9b3cd562676	La Republica 6649	AR	Santa Fe	Rosario	dc902ecb-17ca-4d70-a5bd-d2f09141dff8	null
5	927c7f07-3a9f-4d32-a9eb-146c2c74123f	Campbell 999	AR	Santa Fe	Rosario	8063c9df-23e4-4e58-97d1-9f5b708f61ba	null
6	e7175851-2f5f-41a7-a961-c339b3a5d7f3	Vélez Sársfield 5763	AR	Santa Fe	Rosario	9ae7f5d6-e6ef-46f7-a0b1-2fbfd4c10272	null
7	8118ee0b-0c80-49c2-8489-c4363759ed0	Humberto Primo 1255	AR	Santa Fe	Rosario	935897ae-2dc3-489a-88e9-231ed6fa2bf9	null
8	d6060499-8cbc-4b21-a8b8-cacd17994ed7	Humberto Primo 1255	AR	Santa Fe	Rosario	564c892e-952e-4c41-8959-cdc8aeb653e9	null
9	a9135abe-20e7-4932-9f67-903cceddb94c	Vélez Sársfield 5763	AR	Santa Fe	Rosario	a204c725-1f2b-42ca-8a4a-2c49e979323a	Entrega Copa de leche
10	5a2b543b-5800-4c23-a87d-94e735c108c2	Junín 1836	AR	Santa Fe	Rosario	505f6421-7914-4623-ae09-06055863d79b	null
11	e8b7d961-e650-487b-9fde-ee040cdeb0c3	Pasco 3737	AR	Santa Fe	Rosario	3a795bd0-f1ce-41af-9b82-f2c095c4b374	null
12	9af2076e-22ee-4caa-bc9c-a81cd7b1c2ca	Hutchinson 4241	AR	Santa Fe	Rosario	47ce62bf-c0fb-4118-ae9b-d84803ba31ae	null
13	373358dc-69cc-498b-abad-8532d89e53af	Rio de Janeiro 3437	AR	Santa Fe	Rosario	af3905c9-287e-49a4-9971-dfea687a74f3	null
14	fc7880fd-17a6-42e2-88ee-2b814eb14678	Ecuador 2069	AR	Santa Fe	Rosario	7e0d18a8-6ca5-4dd4-b8d5-308ca4cea12e	null
15	89f163a7-3ec2-48ec-a532-23fe4d1c6105	Riobamba 739	AR	Santa Fe	Rosario	null	Algunas notas
16	130586c8-f371-401a-887e-f759feab286	Av. Provincias Unidas 2050	AR	Santa Fe	Rosario	b9dc3009-9365-42f1-b44f-60049267a18c	null
17	b5db5bf3-7543-4a48-b714-cbb2b25ac05f	Pje. 1112 3317	AR	Santa Fe	Rosario	1b5b205b-d5d3-44bd-864b-64559af0e694	null
18	913d96bf-cff5-45f1-856b-c3e5409a3abd	Maradona Bis 958	AR	Santa Fe	Rosario	70a28379-ae0e-484b-9f42-b2493118b78c	null
19	72fc504b-c760-4d44-9a45-0a82ba7df7f4	Sarratea 1250	AR	Santa Fe	Rosario	1bb09f3d-ba98-40a6-80fe-63b5890d06b4	null
20	fa58dc2-3ced-46d7-ae0b-a72dc944f70	Juan B. Justo 7829	AR	Santa Fe	Rosario	126f325c-9f31-4d45-8d5a-d54a346079de	null
21	f763ee7-02d5-4eda-a9b2-c8f746882be8	San Juan 764	AR	Santa Fe	Rosario	6c37da1e-9d77-48aa-b5bb-2e8503aee41d	null
22	3a98b3a7-9254-400b-8ba3-898d279b99b9	Av. Rivarola 7753	AR	Santa Fe	Rosario	9807f150-8927-4768-b092-8bf83eac61d	null
23	1a509e68-0b2a-470b-a112-d8ffa4fad0d6f	Urquiza 9383	AR	Santa Fe	Rosario	77a2a33f-b271-418c-89e6-580925b93a9d	null
24	91384340-3444-4a9a-a92f-297eb87f7107	Cabal 1252	AR	Santa Fe	Rosario	e7f25756-a306-4cc8-8717-04e729887b32	null
25	c4e325f2-bc10-4566-9ded-ef4294d99606	Chacabuco 4058	AR	Santa Fe	Rosario	6b66f791-c575-4366-b63b-02646bb4c6bb	null

El resultado se guardo en CSV y JSON respectivamente: <https://drive.google.com/file/d/1-GOQr9tykbgBZkp0dAD3YNiaV0Df82Cg/view?usp=sharing>

<https://drive.google.com/file/d/146WwUPf7Kp9l-TEQnOqdaEz648H5zb1M/view?usp=sharing>

• Tabla deliveries

Usaremos la tabla **deliveries_sin_repetidos_tabla** (20 mb total), que tiene la misma estructura de deliveries(40 mb total), pero sin filas repetidas (la primera es de agosto de 2019 y la segunda de mayo de 2020, por lo que asumimos que la segunda es una “evolución” de la primera).

deliveries

[Esquema](#)[Detalles](#)[Vista previa](#)

Nombre del campo	Tipo	Modo	Etiquetas de política 	Descripción
dropoff	STRING	NULLABLE		
trip_id	STRING	NULLABLE		
gateway_data	STRING	NULLABLE		
id	INTEGER	NULLABLE		
extras	STRING	NULLABLE		
_sdc_table_version	INTEGER	NULLABLE		
order_id	STRING	NULLABLE		
updated_at	TIMESTAMP	NULLABLE		
status	STRING	NULLABLE		
_sdc_received_at	TIMESTAMP	NULLABLE		
_sdc_sequence	INTEGER	NULLABLE		
amount	NUMERIC	NULLABLE		
origin_id	STRING	NULLABLE		
created_at	TIMESTAMP	NULLABLE		
pickup	STRING	NULLABLE		
gateway	STRING	NULLABLE		
_sdc_batched_at	TIMESTAMP	NULLABLE		
_sdc_extracted_at	TIMESTAMP	NULLABLE		
destination_id	STRING	NULLABLE		
gateway_id	STRING	NULLABLE		

[Editar esquema](#)

Descripción

Ninguna

Información de la tabla

ID de tabla	nilus-bi:nilus_logistics.deliveries_sin_repetidos_tabla
Tamaño de la tabla	14,11 MB
Número de filas	12.300
Creación	21 may 2020 15:04:33
Caducidad de la tabla	Nunca
Última modificación	20 ene 2021 15:05:34
Ubicación de los datos	US

Exploración inicial de datos:

1	SELECT * FROM `nilus-bi.nilus_logistics.deliveries_sin_repetidos_tabla`
---	---

Resultados de la consulta

 GUARDAR RESULTADOS

 EXAMINAR LOS DATOS 

Se ha completado la consulta (tiempo transcurrido: 0,0 s; almacenada en caché)

Información de la tarea

Resultados

JSON

Fila	dropoff	trip_id
1	{\"notes\": null, \"place\": \"1 -Prueba beneficiario\", \"latlng\": \"20.66487429999999,87.0707602\", \"address\": {\"id\": \"37284428-2c24-4511-877f-1208785f134c\", \"city\": \"La toscana\", \"state\": \"Quintana Roo\", \"country\": \"MX\", \"street_1\": \"Mar de Creta 77\", \"street_2\": null, \"zip_code\": \"54840\", \"telephone\": null}, \"contact\": {\"name\": \"Hernandez\", \"email\": \"hernandez@\", \"cellphone\": \"2235225451\"}, \"open_hours\": {\"friday\": [], \"monday\": [], \"sunday\": [], \"tuesday\": [], \"saturday\": [], \"thursday\": [], \"wednesday\": [\"to\": \"13:00\", \"from\": \"08:00\"]}}	bd9a4095-c197-4303-b21c-c4e6883b442c
2	{\"notes\": \"\", \"place\": \"Hogar el Jardin Floreciente\", \"latlng\": \"18.3988557,-66.1750779\", \"address\": {\"id\": \"aed25037-edeb-4ed0-a308-30ea3f059a13\", \"city\": \"Bayamon\", \"state\": \"Bayamón\", \"country\": \"PR\", \"street_1\": \"Cancha ABB Bayamón (Vaqueros)\", \"street_2\": null, \"zip_code\": \"00959\", \"telephone\": null}, \"contact\": {\"name\": \"Sonia de la Cruz\", \"email\": \"n/a\", \"cellphone\": \"7876975597\"}, \"open_hours\": {\"friday\": [], \"monday\": [], \"sunday\": [], \"tuesday\": [], \"saturday\": [], \"thursday\": [], \"wednesday\": []}}	null
3	{\"notes\": null, \"place\": \"Mesón de Amor\", \"latlng\": \"18.3993851,-66.0514097\", \"address\": {\"id\": \"8490ac36-49da-4143-a5e7-6bf5502a666c\", \"city\": \"San Juan\", \"state\": \"Rio Piedras\", \"country\": \"PR\", \"street_1\": \"1058 Calle Brumbaugh\", \"street_2\": null, \"zip_code\": \"00925\", \"telephone\": null}, \"contact\": {\"name\": \"Belkys Moya\", \"email\": \"mesondeamorr@gmail.com\", \"cellphone\": \"787-376-9704\"}, \"open_hours\": {\"friday\": [], \"monday\": [], \"sunday\": [], \"tuesday\": [], \"saturday\": [], \"thursday\": [], \"wednesday\": []}}	null
4	{\"notes\": null, \"place\": \"Mesón de Amor\", \"latlng\": \"18.3993851,-66.0514097\", \"address\": {\"id\": \"8490ac36-49da-4143-a5e7-6bf5502a666c\", \"city\": \"San Juan\", \"state\": \"Rio Piedras\", \"country\": \"PR\", \"street_1\": \"1058 Calle Brumbaugh\", \"street_2\": null, \"zip_code\": \"00925\", \"telephone\": null}, \"contact\": {\"name\": \"Belkys Moya\", \"email\": \"mesondeamorr@gmail.com\", \"cellphone\": \"787-376-9704\"}, \"open_hours\": {\"friday\": [], \"monday\": [], \"sunday\": [], \"tuesday\": [], \"saturday\": [], \"thursday\": [], \"wednesday\": []}}	null
5	{\"notes\": null, \"place\": \"Mesón de Amor\", \"latlng\": \"18.3993851,-66.0514097\", \"address\": {\"id\": \"8490ac36-49da-4143-a5e7-6bf5502a666c\", \"city\": \"San Juan\", \"state\": \"Rio Piedras\", \"country\": \"PR\", \"street_1\": \"1058 Calle Brumbaugh\", \"street_2\": null, \"zip_code\": \"00925\", \"telephone\": null}, \"contact\": {\"name\": \"Belkys Moya\", \"email\": \"mesondeamorr@gmail.com\", \"cellphone\": \"787-376-9704\"}, \"open_hours\": {\"friday\": [], \"monday\": [], \"sunday\": [], \"tuesday\": [], \"saturday\": [], \"thursday\": [], \"wednesday\": []}}	null
6	{\"notes\": \"\", \"place\": \"La Fondita de Jesús\", \"latlng\": \"18.4505691,-66.0791397\", \"address\": {\"id\": \"bdf0776c-8fcf-4699-8f78-757d74535d10\", \"city\": \"San Juan\", \"state\": \"PR\", \"country\": \"US\", \"street_1\": \"La Fondita de Jesús\", \"street_2\": null, \"zip_code\": \"00907\", \"telephone\": null}, \"contact\": {\"name\": \"Edwin Otero\", \"email\": \"eotero@lafonditadejesus.org\", \"cellphone\": \"0017871415569\"}, \"open_hours\": {\"friday\": [], \"monday\": [], \"sunday\": [], \"tuesday\": [], \"saturday\": [], \"thursday\": [], \"wednesday\": []}}	null
7	{\"notes\": \"\", \"place\": \"La Fondita de Jesús\", \"latlng\": \"18.4505691,-66.0791397\", \"address\": {\"id\": \"bdf0776c-8fcf-4699-8f78-757d74535d10\", \"city\": \"San Juan\", \"state\": \"PR\", \"country\": \"US\", \"street_1\": \"La Fondita de Jesús\", \"street_2\": null, \"zip_code\": \"00907\", \"telephone\": null}, \"contact\": {\"name\": \"Edwin Otero\", \"email\": \"eotero@lafonditadejesus.org\", \"cellphone\": \"0017871415569\"}, \"open_hours\": {\"friday\": [], \"monday\": [], \"sunday\": [], \"tuesday\": [], \"saturday\": [], \"thursday\": [], \"wednesday\": []}}	null
8	{\"notes\": \"\", \"place\": \"Hogar Delia Maria\", \"latlng\": \"18.3858097,-66.1990438\", \"address\": {\"id\": \"d9d36adc-24c8-4298-85ea-064ae0c5e988\", \"city\": \"Mucarabones\", \"state\": \"Toa Alta\", \"country\": \"PR\", \"street_1\": \"Hogar Delia Maria Institución de Envejecientes\", \"street_2\": null, \"zip_code\": \"00953\", \"telephone\": null}, \"contact\": {\"name\": \"Edna Soto\", \"email\": \"0\", \"cellphone\": \"9392061676\"},	null

Filas por página: 100 1 - 100 de 12300

Primera página < >

Por lo que se puede apreciar en una primera vista, los campos más relevantes serían “dropoff” y “pickup”, ambos JSON **idénticos** que contienen la información sobre las direcciones de recogida y entrega de los pedidos.

Al ordenar los datos por el campo “id”, e incluir “created_at” y observar el contenido de los datos se puede apreciar que las operaciones comenzaron en Rosario Argentina para luego ir incorporando a Buenos Aires sobre noviembre de 2019.

De los 12300 registros que tiene la tabla, los primeros 5000 son exclusivamente referidos a las operaciones en Rosario Argentina. En enero de 2020 comienzan con la operación en Puerto Rico, ya para abril de 2020 se incorpora La Plata - Argentina y en mayo del mismo año México.

Campos de “dropoff” y “pickup”:

4620	{ "notes": "", "place": "LA CASA DEL PUEBLO", "latlng": "19.3644012,-99.0028983", "address": { "id": "8110095c-2e67-472b-bd5a-146f1af05621", "city": null, "state": "Ciudad de México", "country": "MX", "street_1": "Comedor Comunitario Casa Del Pueblo", "street_2": null, "zip_code": "", "telephone": null, "contact": { "name": "Alfredo", "email": "-", "cellphone": "+5529174622", "open_hours": { "friday": [], "monday": [{"to": "15:00", "from": "02:00"}], "sunday": [], "tuesday": [], "saturday": [], "thursday": [], "wednesday": []} } }
------	--

Mirando el contenido del campo “country”, por ejemplo se puede ver que se utiliza la misma tabla para los “deliveries” de todos los países donde nilus tiene distribución (PR, MX, US, ..., el nombre del país puede venir con sus dos primeras letras o el nombre completo.).

El campo “notes” puede contener el nombre del punto de delivery como así alguna indicación para su más rápida ubicación por parte de los choferes. También el campo “place” se utiliza para guardar el nombre del punto de delivery. Solo mirando los datos no es posible darse cuenta de cuando usan uno y/u otro. Pueden estar ambos vacíos o llenos en cualquier combinación y como se ve en la imagen arriba, “street_1” puede no contener estrictamente el nombre de la calle.

1267	{ "notes": "Teniente agneta y las vias", "place": "Club Padre Edgardo Montalo", "latlng": "-32.930"
------	---

Otro campo relevante es “extras” donde se guarda información referida a si necesita o no “refrigeración” por ej. (ojo, solo dice “refrigerated” sin especificar tiempo mínimo que puede estar sin ella o temperatura mínima necesaria o recomendable). No hay muchas filas que tengan este dato cargado, pero por la fecha de alguno que sí está cargado parece que se está utilizando

El campo “amount” solo tiene valores diferentes de cero para 5794 filas de las 12300 totales.

```
1 SELECT * FROM `nilus-bi.nilus_logistics.deliveries_sin_repetidos_tabla` order by id
```

Resultados de la consulta [GUARDAR RESULTADOS](#) [EXAMINAR LOS DATOS](#) ▼

Se ha completado la consulta (tiempo transcurrido: 1,3 s; bytes procesados: 14,1 MB)

Información de la tarea [Resultados](#) [JSON](#) [Detalles de ejecución](#)

gateway_data	id	extras	_sdc_table_version	order_id	updated_at	status	_sdc_received_at	_sdc_sequence
{}	12196	{'options': []}	1565122950921	88b6a364-bbf3-43e8-b4fe-525cc5a63908	2020-11-09 20:47:13.062 UTC	processing	2020-11-10 17:22:23.037 UTC	1605028942833162001
{}	12197	{'options': ['refrigerated']}	1565122950921	89faac10-ae36-4843-8e1f-6a00993b7c2f	2020-11-10 16:51:46.680 UTC	processing	2020-11-11 23:22:22.849 UTC	1605136942622905001

```
1 SELECT * FROM `nilus-bi.nilus_logistics.deliveries_sin_repetidos_tabla` where amount <> 0 order by id
```

	status	_sdc_received_at	_sdc_sequence	amount	origin_id
829 UTC	completed	2019-08-06 20:22:32.328 UTC	1565122951534000048	250	fb29f78b-90
743 UTC	completed	2019-08-06 20:22:32.328 UTC	1565122951534000095	50	fb29f78b-90
855 UTC	completed	2019-08-06 20:22:32.328 UTC	1565122951535000100	250	fb29f78b-90

Para el campo “Trip_id solo hay valores para 4723 filas.

```
SELECT * FROM `nilus-bi.nilus_logistics.deliveries_sin_repetidos_tabla` where trip_id <> "" order by id
```

Para el caso de la tabla deliveries el trip_id puede venir sin cargar tanto si el estatus es "processing", "canceled" o "assigned" y cuando no viene vacío los estados pueden ser "assigned", "completed" o "Canceled.

```
SELECT distinct(status) FROM `nilus-bi.nilus_logistics.deliveries_sin_repetidos_tabla`
where trip_id is null
```

Resultados de la consulta		
Se ha completado la consulta (tiempo trans		
Información de la tarea		Resultados
Fila	status	
1	processing	
2	canceled	
3	assigned	

SELECT distinct(status) FROM `nilus-bi.nilus\_logistics.deliveries\_sin\_repetidos\_tabla`
where trip_id is not null

Resultados de la consulta		
Se ha completado la consulta (tiempo trans		
Información de la tarea		Resultados
Fila	status	
1	assigned	
2	completed	
3	canceled	

Versión completa de la tabla extraída como CSV en:

Versión completa de la tabla extraída como JSON en:

https://drive.google.com/file/d/1hSCaFnNG9lpfe1ZzaA16TG_H9gnrGmac/view?usp=sharing

Selección final y extracción de campos relevantes de la tabla “deliveries_sin_repetidos”

```
SELECT created_at, id, order_id, trip_id, pickup, dropoff, extras, status, amount, origin_id, destination_id FROM `nilus-bi.nilus_logistics.deliveries_sin_repetidos_tabla` order by id
```

Extracción completa de campos seleccionados relevantes:

CSV

https://drive.google.com/file/d/1W_UQ7OcW0l7_SGsQFTbGCSs8rvBaCn8C/view?usp=sharing

JSON

<https://drive.google.com/file/d/1dzHfZuCOMibBRIXW2CN5KqwgjbV51VaJ/view?usp=sharing>

deliveries_sin_repetidos_tabla

[Esquema](#)
[Detalles](#)
[Vista previa](#)

Nombre del campo	Tipo	Modo
dropoff	STRING	NULLABLE
trip_id	STRING	NULLABLE
gateway_data	STRING	NULLABLE
id	INTEGER	NULLABLE
extras	STRING	NULLABLE
_sdc_table_version	INTEGER	NULLABLE
order_id	STRING	NULLABLE
updated_at	TIMESTAMP	NULLABLE
status	STRING	NULLABLE
_sdc_received_at	TIMESTAMP	NULLABLE
_sdc_sequence	INTEGER	NULLABLE
amount	NUMERIC	NULLABLE
origin_id	STRING	NULLABLE
created_at	TIMESTAMP	NULLABLE
pickup	STRING	NULLABLE
gateway	STRING	NULLABLE
_sdc_batched_at	TIMESTAMP	NULLABLE
_sdc_extracted_at	TIMESTAMP	NULLABLE
destination_id	STRING	NULLABLE
gateway_id	STRING	NULLABLE

- **Tabla Milestones**

milestones

Esquema Detalles Vista previa

Nombre del campo	Tipo	Modo
trip_id	STRING	NULLABLE
id	INTEGER	NULLABLE
name	STRING	NULLABLE
_sdc_table_version	INTEGER	NULLABLE
network_id	STRING	NULLABLE
_sdc_received_at	TIMESTAMP	NULLABLE
_sdc_sequence	INTEGER	NULLABLE
created_at	TIMESTAMP	NULLABLE
_sdc_batched_at	TIMESTAMP	NULLABLE
data	STRING	NULLABLE
_sdc_extracted_at	TIMESTAMP	NULLABLE

Esquema Detalles Vista previa

Descripción

Ninguna

Información de la tabla

ID de tabla	nilus-bi:nilus_logistics.milestones
Tamaño de la tabla	2,38 MB
Número de filas	21.474
Creación	6 ago 2019 17:46:52
Caducidad de la tabla	Nunca
Última modificación	15 dic 2020 17:33:20
Ubicación de los datos	US

Esquema	Detalles	Vista previa									
Fila	trip_id	id	name	_sdc_table_version	network_id	_sdc_received_at	_sdc_sequence	created_at	_sdc_batched_at	data	_sdc_extracted_at
21401	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 00:22:22.299 UTC	1607818942085845001	2020-11-02 22:34:31.681 UTC	2020-12-13 00:30:04.390 UTC	0	2020-12-13 00:22:21.827 UTC
21402	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-12 15:22:23.080 UTC	1607786542876120001	2020-11-02 22:34:31.681 UTC	2020-12-12 15:31:01.323 UTC	0	2020-12-12 15:22:22.713 UTC
21403	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-12 18:22:23.680 UTC	1607793743395285001	2020-11-02 22:34:31.681 UTC	2020-12-12 18:31:40.546 UTC	0	2020-12-12 18:22:23.093 UTC
21404	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-12 17:22:23.180 UTC	1607793742873148001	2020-11-02 22:34:31.681 UTC	2020-12-12 17:31:47.438 UTC	0	2020-12-12 17:22:22.657 UTC
21405	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-12 16:22:23.328 UTC	1607790143038096001	2020-11-02 22:34:31.681 UTC	2020-12-12 16:31:54.349 UTC	0	2020-12-12 16:22:22.738 UTC
21406	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-12 19:22:21.936 UTC	1607800941804945001	2020-11-02 22:34:31.681 UTC	2020-12-12 19:31:31.140 UTC	0	2020-12-12 19:22:21.633 UTC
21407	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 05:22:24.014 UTC	1607836943272010001	2020-11-02 22:34:31.681 UTC	2020-12-13 05:28:58.940 UTC	0	2020-12-13 05:22:22.276 UTC
21408	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 04:22:22.905 UTC	1607833342798747001	2020-11-02 22:34:31.681 UTC	2020-12-13 04:29:09.462 UTC	0	2020-12-13 04:22:22.665 UTC
21409	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 03:22:23.254 UTC	1607829743020715001	2020-11-02 22:34:31.681 UTC	2020-12-13 03:29:20.922 UTC	0	2020-12-13 03:22:22.695 UTC
21410	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 02:22:23.300 UTC	1607826143045896001	2020-11-02 22:34:31.681 UTC	2020-12-13 02:29:34.434 UTC	0	2020-12-13 02:22:22.733 UTC
21411	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 01:22:22.026 UTC	1607822541813078001	2020-11-02 22:34:31.681 UTC	2020-12-13 01:29:47.921 UTC	0	2020-12-13 01:22:21.359 UTC
21412	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 13:22:23.069 UTC	160786574272185001	2020-11-02 22:34:31.681 UTC	2020-12-13 13:27:59.839 UTC	0	2020-12-13 13:22:22.496 UTC
21413	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 11:22:23.034 UTC	1607858542810821001	2020-11-02 22:34:31.681 UTC	2020-12-13 11:28:14.934 UTC	0	2020-12-13 11:22:22.498 UTC
21414	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 12:22:22.163 UTC	1607862142050902001	2020-11-02 22:34:31.681 UTC	2020-12-13 12:28:08.037 UTC	0	2020-12-13 12:22:21.845 UTC
21415	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 14:22:22.982 UTC	1607869342743593001	2020-11-02 22:34:31.681 UTC	2020-12-13 14:27:52.039 UTC	0	2020-12-13 14:22:22.367 UTC
21416	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 15:22:23.228 UTC	1607872942935703001	2020-11-02 22:34:31.681 UTC	2020-12-13 15:27:44.663 UTC	0	2020-12-13 15:22:22.547 UTC
21417	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 07:22:23.321 UTC	1607844143013988001	2020-11-02 22:34:31.681 UTC	2020-12-13 07:28:41.829 UTC	0	2020-12-13 07:22:22.738 UTC
21418	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 06:22:21.788 UTC	1607840541572525001	2020-11-02 22:34:31.681 UTC	2020-12-13 06:28:49.620 UTC	0	2020-12-13 06:22:21.374 UTC
21419	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 10:22:23.532 UTC	1607854943268110001	2020-11-02 22:34:31.681 UTC	2020-12-13 10:28:21.781 UTC	0	2020-12-13 10:22:23.113 UTC
21420	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 09:22:23.697 UTC	1607851342574206001	2020-11-02 22:34:31.681 UTC	2020-12-13 09:28:28.540 UTC	0	2020-12-13 09:22:22.213 UTC
21421	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 08:22:22.966 UTC	1607847742777637001	2020-11-02 22:34:31.681 UTC	2020-12-13 08:28:34.923 UTC	0	2020-12-13 08:22:22.640 UTC
21422	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 16:22:23.066 UTC	1607876542839581001	2020-11-02 22:34:31.681 UTC	2020-12-13 16:27:41.166 UTC	0	2020-12-13 16:22:22.639 UTC
21423	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 20:22:21.970 UTC	1607890941800183001	2020-11-02 22:34:31.681 UTC	2020-12-13 20:26:57.040 UTC	0	2020-12-13 20:22:21.709 UTC
21424	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 19:22:23.203 UTC	1607887343057280001	2020-11-02 22:34:31.681 UTC	2020-12-13 19:27:07.445 UTC	0	2020-12-13 19:22:22.897 UTC
21425	21998e59-8f8f-4183-b8fb-7bf5590e391f	9769	completed	1565122952997	MCBA	2020-12-13 18:22:21.753 UTC	1607883741493985001	2020-11-02 22:34:31.681 UTC	2020-12-13 18:27:18.024 UTC	0	2020-12-13 18:22:21.411 UTC

Elas por página: 100

21401 - 21474 de 21474

Primera página

Filas por página: 100 21401 - 21474 de 21474 Primera página 16

Versión completa de la tabla extraída como CSV en:

Versión completa de la tabla extraída como JSON en:

<https://drive.google.com/file/d/1TRNiP5ERh2BUOShYwJYfv-hgjfCTdyBT/view?usp=sharing>

Selección final y extracción de campos relevantes de la tabla “milestones”

```
SELECT id, trip_id, name, data, created_at FROM `nilus-bi.nilus_logistics.milestones` order by trip_id
```

milestones

Esquema Detalles Vista previa

Nombre del campo	Tipo	Modo
trip_id	STRING	NULLABLE
id	INTEGER	NULLABLE
name	STRING	NULLABLE
_sdc_table_version	INTEGER	NULLABLE
network_id	STRING	NULLABLE
_sdc_received_at	TIMESTAMP	NULLABLE
_sdc_sequence	INTEGER	NULLABLE
created_at	TIMESTAMP	NULLABLE
_sdc_batched_at	TIMESTAMP	NULLABLE
data	STRING	NULLABLE
_sdc_extracted_at	TIMESTAMP	NULLABLE

Resultados de la consulta

[GUARDAR RESULTADOS](#)

[EXAMINAR LOS DATOS](#) ▼

Se ha completado la consulta (tiempo transcurrido: 1,1 s; bytes procesados: 1,5 MB)

Información de la tarea Resultados JSON Detalles de ejecución

22476	691	ff1647ce-370a-4999-a589-b6916bcc4a7f	heading_to_dropoff_4	{}	2018-08-31 14:37:08.475 UTC
22477	692	ff1647ce-370a-4999-a589-b6916bcc4a7f	at_dropoff_4	{}	2018-08-31 14:37:11.583 UTC
22478	693	ff1647ce-370a-4999-a589-b6916bcc4a7f	completed	{}	2018-08-31 14:37:12.845 UTC
22479	694	ff1647ce-370a-4999-a589-b6916bcc4a7f	completed	{}	2018-08-31 14:37:14.093 UTC
22480	3429	ff30eb1c-521f-40c1-9995-86f9aa51fdf8	heading_to_dropoff_0	{}	2019-03-26 18:42:23.291 UTC
22481	3430	ff30eb1c-521f-40c1-9995-86f9aa51fdf8	init	{}	2019-03-26 18:47:11.886 UTC
22482	3431	ff30eb1c-521f-40c1-9995-86f9aa51fdf8	at_pickup_0	{}	2019-03-26 18:47:16.367 UTC
22483	3432	ff30eb1c-521f-40c1-9995-86f9aa51fdf8	heading_to_dropoff_0	{}	2019-03-26 18:47:19.517 UTC
22484	3441	ff30eb1c-521f-40c1-9995-86f9aa51fdf8	heading_to_dropoff_1	{}	2019-03-27 15:04:11.343 UTC
22485	3442	ff30eb1c-521f-40c1-9995-86f9aa51fdf8	at_dropoff_1	{}	2019-03-27 15:04:12.315 UTC
22486	3443	ff30eb1c-521f-40c1-9995-86f9aa51fdf8	completed	{}	2019-03-27 15:04:13.274 UTC
22487	3444	ff30eb1c-521f-40c1-9995-86f9aa51fdf8	completed	{}	2019-03-27 15:04:14.383 UTC
22488	2856	ffa724ab-8a5c-4621-859d-670a17ec2150	init	{}	2019-02-19 12:20:33.953 UTC
22489	2857	ffa724ab-8a5c-4621-859d-670a17ec2150	at_pickup_0	{}	2019-02-19 12:20:38.437 UTC
22490	2858	ffa724ab-8a5c-4621-859d-670a17ec2150	heading_to_dropoff_0	{}	2019-02-19 12:20:39.770 UTC
22491	2859	ffa724ab-8a5c-4621-859d-670a17ec2150	at_dropoff_0	{}	2019-02-19 12:20:40.904 UTC
22492	2860	ffa724ab-8a5c-4621-859d-670a17ec2150	completed	{}	2019-02-19 12:20:41.964 UTC
22493	2861	ffa724ab-8a5c-4621-859d-670a17ec2150	completed	{}	2019-02-19 12:20:43.506 UTC

Esta tabla está actualizada solo hasta febrero de 2019, habría que verificar que la sigan utilizando.

El campo “data” lo utilizaron alguna vez para guardar el estado final aparentemente, pues solo hay 38 de las 22493 filas con algún dato cargado.

```
SELECT id, trip_id, name, data, created_at FROM `nilus-bi.nilus_logistics.milestones` where data <> '{}' order by trip_id
```

Resultados de la consulta

GUARDAR RESULTADOS

EXAMINAR LOS DATOS

Se ha completado la consulta (tiempo transcurrido: 0,3 s; bytes procesados: 1,5 MB)

Información de la tarea

Resultados

JSON

Detalles de ejecución

17	9790	21235af6-5f4c-4ce0-9cff-0a5f1cbf78a5	heading_to_dropoff_4	("status": "delivered", "delivery_id": 12612)	2021-01-15 11:57:37.384 UTC
18	9792	21235af6-5f4c-4ce0-9cff-0a5f1cbf78a5	heading_to_dropoff_5	("status": "no_delivered", "delivery_id": 12613)	2021-01-15 11:58:33.224 UTC
19	9794	21235af6-5f4c-4ce0-9cff-0a5f1cbf78a5	heading_to_dropoff_6	("status": "delivered", "delivery_id": 12614)	2021-01-15 11:58:37.047 UTC
20	9796	21235af6-5f4c-4ce0-9cff-0a5f1cbf78a5	completed	("status": "delivered", "delivery_id": 12615)	2021-01-15 11:58:41.011 UTC
21	9802	35a8d34d-6661-4520-b056-4e9e2760906e	heading_to_dropoff_1	("status": "delivered", "delivery_id": 12645)	2021-01-15 12:00:15.313 UTC
22	9804	35a8d34d-6661-4520-b056-4e9e2760906e	heading_to_dropoff_2	("status": "no_delivered", "delivery_id": 12646)	2021-01-15 12:00:22.989 UTC
23	9806	35a8d34d-6661-4520-b056-4e9e2760906e	heading_to_dropoff_3	("status": "delivered", "delivery_id": 12647)	2021-01-15 12:00:46.654 UTC
24	9808	35a8d34d-6661-4520-b056-4e9e2760906e	heading_to_dropoff_4	("status": "delivered", "delivery_id": 12649)	2021-01-15 12:00:54.899 UTC
25	9810	35a8d34d-6661-4520-b056-4e9e2760906e	heading_to_dropoff_5	("status": "delivered", "delivery_id": 12650)	2021-01-15 12:00:59.696 UTC
26	9812	35a8d34d-6661-4520-b056-4e9e2760906e	heading_to_dropoff_6	("status": "delivered", "delivery_id": 12651)	2021-01-15 12:01:05.100 UTC

Filas por página: 1001 - 38 de 38

Versión completa de los campos seleccionados de “milestones” como JSON en: <https://drive.google.com/file/d/1THu5O7j3zINzUkge5NS6RDS5v52FuT4S/view?usp=sharing>

- **Tabla Vehicles**

vehicles		
Esquema	Detalles	Vista previa
Nombre del campo	Tipo	Modo
model	STRING	NULLABLE
brand	STRING	NULLABLE
id	STRING	NULLABLE
extras	STRING	NULLABLE
_sdc_table_version	INTEGER	NULLABLE
updated_at	TIMESTAMP	NULLABLE
shipper_id	STRING	NULLABLE
network_id	STRING	NULLABLE
_sdc_received_at	TIMESTAMP	NULLABLE
_sdc_sequence	INTEGER	NULLABLE
year	INTEGER	NULLABLE
created_at	TIMESTAMP	NULLABLE
_sdc_batched_at	TIMESTAMP	NULLABLE
_sdc_extracted_at	TIMESTAMP	NULLABLE

Esquema	Detalles	Vista previa
---------	----------	--------------

Descripción

Ninguna

Información de la tabla

ID de tabla	nilus-bi:nilus_logistics.vehicles
Tamaño de la tabla	2,24 MB
Número de filas	11.778
Creación	6 ago 2019 17:45:55
Caducidad de la tabla	Nunca
Última modificación	15 dic 2020 17:33:04
Ubicación de los datos	US

Agrupar													
Agrupar		Detalles		Vista previa									
11706	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-12 15:22:24.256 UTC	160778654409899001	2020	2020-12-11 21:12:28.317 UTC	2020-12-12 15:1
11707	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-12 16:22:24.405 UTC	16077901442765630001	2020	2020-12-11 21:12:28.317 UTC	2020-12-12 16:1
11708	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-12 12:22:25.582 UTC	16077757455272320001	2020	2020-12-11 21:12:28.317 UTC	2020-12-12 12:1
11709	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 02:22:24.306 UTC	160782616440531820001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 02:1
11710	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 01:22:22.958 UTC	16078225428436870001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 01:1
11711	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-12 23:22:23.741 UTC	16078153434650110001	2020	2020-12-11 21:12:28.317 UTC	2020-12-12 23:1
11712	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 03:22:24.290 UTC	16078297439776950001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 03:1
11713	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 00:22:23.005 UTC	16078199429247950001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 00:1
11714	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 16:22:24.057 UTC	16078765439147760001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 16:1
11715	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 14:22:24.073 UTC	16078693438556190001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 14:1
11716	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 17:22:23.862 UTC	16078801437077460001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 17:1
11717	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 15:22:23.831 UTC	16078729438060280001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 15:1
11718	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 18:22:22.628 UTC	16078837424613870001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 18:1
11719	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 07:22:24.133 UTC	16078441439958960001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 07:1
11720	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 06:22:22.532 UTC	16078405424470230001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 06:1
11721	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 04:22:23.673 UTC	1607833464514190001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 04:1
11722	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 05:22:24.773 UTC	16078369445995280001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 05:1
11723	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 08:22:23.980 UTC	160784774373368110001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 08:1
11724	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 23:22:23.930 UTC	16079017437312950001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 23:1
11725	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 22:22:22.744 UTC	16078981426372340001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 22:1
11726	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 21:22:23.904 UTC	16078945437524900001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 21:1
11727	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 20:22:22.823 UTC	16078909427387160001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 20:1
11728	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 19:22:24.272 UTC	16078673441337910001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 19:1
11729	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 11:22:23.676 UTC	16078585435618300001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 11:1
11730	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 09:22:23.819 UTC	16078513436741480001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 09:1
11731	toyota	corola	db2499ae-2ae6-4cdf-a837-ab79b6f1dbd7	['Features', 'refrigerated']	1565122968357	2020-12-11 21:12:28.317 UTC	57056fae-b432-4296-936f-4138f02c7dbc	MCBA	2020-12-13 10:22:24.515 UTC	16078549442536390001	2020	2020-12-11 21:12:28.317 UTC	2020-12-13 10:1

Filas por página: 100

11701 - 11778 de 11778

Primera página

<

>

Última página

Versión completa de la tabla extraída como JSON en:

https://drive.google.com/file/d/1RkIEWh4jYnF_pU_aLchTlibggoAVp1aR/view?usp=sharing

SELECT count(brand), brand FROM `nilus-bi.nilus\_logistics.vehicles` group by (brand)

Resultados de la consulta

Se ha completado la consulta (tiempo transcurr

Información de la tarea [Resultados](#) JS

Fila	f0_	brand	
1	243	Fiat	
2	7067	Mitsubishi	
3	773	Peugeot	
4	3	Iveco - Pat:KRJ054	
5	0	<i>null</i>	
6	4	Renault	
7	1	Chevrolet	
8	1	Mercedes Benz	
9	1	MB	
10	1	VW	
11	1	Ford	
12	1	FORD	
13	6	Mercedes Benz	
14	22	Citroen	
15	3550	Ford Patente JPQ912	
16	1031	corola	

```
SELECT distinct(shipper_id), id, shipper_id, year, model, brand FROM `nilus-  
bi.nilus_logistics.vehicles`
```

Resultados de la consulta				 GUARDAR RESULTADOS	 EXAMINAR LOS DATOS 	
Se ha completado la consulta (tiempo transcurrido: 0,5 s; bytes procesados: 1,4 MB)						
Información de la tarea		Resultados	JSON	Detalles de ejecución		
8	202e44d6-9487-4fd6-a8cb-8bc2aea6e44b	fe05e543-c59e-4e5b-87ca-296e9623ba4e	202e44d6-9487-4fd6-a8cb-8bc2aea6e44b	2017	Volkswagen - Suran 1.6	null
9	d6cf99f3-ee63-4722-944b-f6b915686cd6	d7bee9ea-7330-47ca-9a71-d28e5f93d9cf	d6cf99f3-ee63-4722-944b-f6b915686cd6	0	-	null
10	c25a9593-7a2b-4e0a-937b-a2ed7fc30cdb	a51cd2dd-4abc-404e-a319-6ffc3c4c3795	c25a9593-7a2b-4e0a-937b-a2ed7fc30cdb	2006	Pick up Chevrolet S10 2.8 TDI STD 4x2 ca	null
11	ac1dad18-7f20-4e03-8a35-f3b2db644e73	60e00748-7514-4742-9728-6fde44ff8097	ac1dad18-7f20-4e03-8a35-f3b2db644e73	0	Test Type	null
12	bb99e1ad-fb0f-40ab-af44-50a153b4b20f	78ed4ba6-18a4-40d4-8b42-35a997130bc6	bb99e1ad-fb0f-40ab-af44-50a153b4b20f	0	-	null
13	6713e7a9-c2d5-48d8-b714-b04c22bf20a8	ad7cc8cb-b69f-44f9-a74e-c3b41b44a20	6713e7a9-c2d5-48d8-b714-b04c22bf20a8	0	Tipo	null
14	d73a1286-49af-4638-a1c9-98733f3bb9e0	8a69e5d7-06dd-4494-81a6-1ff1aa6c76f4	d73a1286-49af-4638-a1c9-98733f3bb9e0	1979	Volkswagen - Kombi 1.6	null
15	bf7ea961-f335-482d-8eef-349a6a84c577	ff4cce8a-933e-4787-abc4-d5dc53998a38	bf7ea961-f335-482d-8eef-349a6a84c577	2011	Kangoo confort 1.6 5A CD SVT 600 K	Renault
16	206bbb32-9bdc-41d5-8b36-5b7f2fb74278	75c62a20-c3b8-4a13-a8b0-b27ee85a5d88	206bbb32-9bdc-41d5-8b36-5b7f2fb74278	1993	Pick up C-20 custom	Chevrolet
17	9e1269d9-c64c-4170-9f25-f9b582d6d0ef	51def334-2aa3-4c2b-abdb-da2dbc7d93a5	9e1269d9-c64c-4170-9f25-f9b582d6d0ef	2017	Fiorino	Fiat
18	a43b2a9b-e088-4cc3-b7ca-a31635a0aaa3	b797e0fa-035d-45c7-b959-836da5dae7bd	a43b2a9b-e088-4cc3-b7ca-a31635a0aaa3	2010	710	Mercedes Benz
19	a1c79f45-6f13-4a14-8638-4b9801164ffc	c2c84c99-9012-43d2-bb84-b6cba62a5b57	a1c79f45-6f13-4a14-8638-4b9801164ffc	null	Partner	Peugeot
20	19700ea8-e20a-4ff1-b17b-2b1a5c5ed9e0	acd95e9a-c498-4406-9768-04449d148cbb	19700ea8-e20a-4ff1-b17b-2b1a5c5ed9e0	2011	Sprinter	MB
21	adcc04bc-671f-4740-aea5-13ddc0466b0c	f9305c85-393d-4e7f-8319-5bbfe78971ae	adcc04bc-671f-4740-aea5-13ddc0466b0c	2018	Amarok	VW
22	2262689f-6a08-4eda-bf33-2a9a313672f8	c5a9b9f9-337d-47e0-9756-5f6e3352dba6	2262689f-6a08-4eda-bf33-2a9a313672f8	0	-	null
23	f16eeba2-bb55-4d71-a0e4-e71cb6a0c3d0	9d13b461-b345-43d6-9161-a462379f7626	f16eeba2-bb55-4d71-a0e4-e71cb6a0c3d0	null	Master	Renault
24	df1f3847-7fc5-4b99-bf96-7b22433379cf	b7ea61ff-31c9-48ec-b1e8-bc233547408b	df1f3847-7fc5-4b99-bf96-7b22433379cf	null	F100	Ford
25	53e347f5-1797-4266-b807-8ab649d9301b	70c0c44f-a357-4e87-9c78-fad547a6d392	53e347f5-1797-4266-b807-8ab649d9301b	2014	Master PH3 DCI 120 DKCNF	Renault
				Filas por página:	100	1 - 39 de 39

SELECT distinct(id), id, shipper_id, year, model, brand, extras, created_at, updated_at FROM `nilus-bi.nilus\_logistics.vehicles`

Resultados de la consulta							
Se ha completado la consulta (tiempo transcurrido: 0,4 s; bytes procesados: 1,9 MB)							
Información de la tarea		Resultados	JSON	Detalles de ejecución			
Fila	id	id_1	shipper_id	year	model	brand	extras
1	e3353e5f-8a68-4672-96c7-48f5c55f4561	e3353e5f-8a68-4672-96c7-48f5c55f4561	273cd6a1-7c64-4991-bbd6-93bef4d340d0	2014	Ducato	Fiat	({"features": []})
2	5fcd6e3b-7b84-4730-8e81-08e83ae49569	5fcd6e3b-7b84-4730-8e81-08e83ae49569	99e5cf7a-0015-4347-ba19-29626023e798	2014	Mitsubishi Mirage	Mitsubishi	({"features": []})
3	da9d1b43-d1d5-4d23-a94e-dc7ae52ac583	da9d1b43-d1d5-4d23-a94e-dc7ae52ac583	36503a13-289d-4bad-ba7b-07c1259f862f	2008	Partner	Peugeot	({"features": []})
4	5b700890-00d9-489d-8775-614984eb4c0e	5b700890-00d9-489d-8775-614984eb4c0e	36503a13-289d-4bad-ba7b-07c1259f862f	2008	Partner	Peugeot	({"features": []})
5	0bf52511-2e17-4068-b661-10004d0f041a	0bf52511-2e17-4068-b661-10004d0f041a	0b862f61-b87d-4624-b08b-409b1ccbe155	null	Only-SSC14	Iveco - PatKRJ054	({"features": []})
6	0a830f6c-10f2-4834-a1bf-c2a9b9d7f73e	0a830f6c-10f2-4834-a1bf-c2a9b9d7f73e	0b862f61-b87d-4624-b08b-409b1ccbe155	null	Only-SSC14	Iveco - PatKRJ054	({"features": []})
7	03a02d04-af06-4a84-ba4c-c92090177b7e	03a02d04-af06-4a84-ba4c-c92090177b7e	0b862f61-b87d-4624-b08b-409b1ccbe155	2000	Only-SSC14	Iveco - PatKRJ054	({"features": []})
8	fe05e543-c59e-4e5b-87ca-296e9623ba4e	fe05e543-c59e-4e5b-87ca-296e9623ba4e	202e44d6-9487-4fd6-a8cb-8bc2aea6e44b	2017	Volkswagen - Suran 1.6	null	({"capacity": 5, "gateway_id": 6368})
9	d7bee9ea-7330-47ca-9a71-d28e5f93d9cf	d7bee9ea-7330-47ca-9a71-d28e5f93d9cf	d6cf99f3-ee63-4722-944b-f6b915686cd6	0	-	null	({"capacity": 3, "gateway_id": 10360})
10	a51cd2dd-4abc-404e-a319-6ffc3c4c3795	a51cd2dd-4abc-404e-a319-6ffc3c4c3795	c25a9593-7a2b-4e0a-937b-a2ed7fc30cdb	2006	Pick up Chevrolet S10 2.8 TDI STD 4x2 ca	null	({"capacity": 5, "gateway_id": 9142})
11	60e00748-7514-4742-9728-6fde44ff8097	60e00748-7514-4742-9728-6fde44ff8097	ac1dad18-7f20-4e03-8a35-f3b2db644e73	0	Test Type	null	({"capacity": 3, "gateway_id": 4177})
12	78ed4ba6-18a4-40d4-8b42-35a997130bc6	78ed4ba6-18a4-40d4-8b42-35a997130bc6	bb99e1ad-fb0f-40ab-af44-50a153b4b20f	0	-	null	({"capacity": 5, "gateway_id": 8749})
13	ad7cc8cb-b69f-44f9-a74e-c3b41b44a20	ad7cc8cb-b69f-44f9-a74e-c3b41b44a20	6713e7a9-c2d5-48d8-b714-b04c22bf20a8	0	Tipo	null	({"capacity": 3, "gateway_id": 6367})
14	8a69e5d7-06dd-4494-81a6-1ff1aa6c76f4	8a69e5d7-06dd-4494-81a6-1ff1aa6c76f4	d73a1286-49af-4638-a1c9-98733f3bb9e0	1979	Volkswagen - Kombi 1.6	null	({"capacity": 5, "gateway_id": 10365})
15	ff4cce8a-933e-4787-abc4-d5dc53998a38	ff4cce8a-933e-4787-abc4-d5dc53998a38	bf7ea961-f335-482d-8eef-349a6a84c577	2011	Kangoo confort 1.6 5A CD SVT 600 K	Renault	({"capacity": 4, "features": [], "gateway_id": 9037})
16	75c62a20-c3b8-4a13-a8b0-b27ee85a5d88	75c62a20-c3b8-4a13-a8b0-b27ee85a5d88	206bbb32-9bdc-41d5-8b36-5b7f2fb74278	1993	Pick up C-20 custom	Chevrolet	({"capacity": 5, "features": ["refrigerated"], "gateway_id": 91
17	51def334-2aa3-4c2b-abdb-da2dbc7d93a5	51def334-2aa3-4c2b-abdb-da2dbc7d93a5	9e1269d9-c64c-4170-9f25-f9b582d6d0ef	2017	Fiorino	Fiat	({"capacity": 5, "features": ["refrigerated"], "gateway_id": 91
				Filas por página: 100 1 - 39 de 39 Primera página < > > Última página			

Selección final y extracción de campos relevantes de la tabla “vehicles”

https://drive.google.com/file/d/15a5KG1Yo8S8HgSl8o0vxc0slz_npnQl-/view?usp=sharing

vehicles

[Esquema](#) [Detalles](#) [Vista previa](#)

Nombre del campo	Tipo
model	STRING
brand	STRING
id	STRING
extras	STRING
_sdc_table_version	INTEGER
updated_at	TIMESTAMP
shipper_id	STRING
network_id	STRING
_sdc_received_at	TIMESTAMP
_sdc_sequence	INTEGER
year	INTEGER
created_at	TIMESTAMP
_sdc_batched_at	TIMESTAMP
_sdc_extracted_at	TIMESTAMP

- **Tabla Verifications**

(posible candidata a Vehicles dada su actualización y tamaño)

verifications

[Esquema](#)[Detalles](#)[Vista previa](#)

Nombre del campo	Tipo
verifiable_type	STRING
id	INTEGER
_sdc_table_version	INTEGER
updated_at	TIMESTAMP
network_id	STRING
_sdc_received_at	TIMESTAMP
_sdc_sequence	INTEGER
created_at	TIMESTAMP
verifiable_id	STRING
_sdc_batched_at	TIMESTAMP
data	STRING
_sdc_extracted_at	TIMESTAMP

```
SELECT id, created_at, verifiable_id, verifiable_type, data, updated_at FROM `nilus-bi.nilus_logistics.verifications` order by id, verifiable_id
```

Fila	id	created_at	verificable_id	verificable_type	data	updated_at
1	1	2018-02-23 16:26:46.905 UTC	866a4578-fa44-4519-92d4-0f5c5634f965	Vehicle	{"type": "license_plate", "information": {"number": "WUY733"}}	2018-11-23 22:39:54.943 UTC
2	2	2018-02-23 16:26:47.053 UTC	fe05e543-c59e-4e5b-87ca-296e9623ba4e	Vehicle	{"type": "license_plate", "information": {"number": "AGE377"}}	2018-11-23 22:39:54.953 UTC
3	3	2018-02-23 16:26:47.203 UTC	d7bee9ea-7330-47ca-9a71-d28e5f93d9cf	Vehicle	{"type": "license_plate", "information": {"number": "12"}}	2018-11-23 22:39:54.959 UTC
4	4	2018-02-23 16:26:47.493 UTC	a51cd2dd-4abc-404e-a319-6ffc3c4c3795	Vehicle	{"type": "license_plate", "information": {"number": "FKC634"}}	2018-11-23 22:39:54.967 UTC
5	5	2018-02-23 16:26:47.782 UTC	60e00748-7514-4742-9728-6fde44ff8097	Vehicle	{"type": "license_plate", "information": {"number": "12345123"}}	2018-11-23 22:39:54.975 UTC
6	6	2018-02-23 16:26:47.945 UTC	c5a9b9f9-337d-47e0-9756-5f6e3352dba6	Vehicle	{"type": "license_plate", "information": {"number": "-."}}	2018-11-23 22:39:54.982 UTC
7	7	2018-02-23 16:26:48.104 UTC	78ed4ba6-18a4-40d4-8b42-35a997130bc6	Vehicle	{"type": "license_plate", "information": {"number": "-."}}	2018-11-23 22:39:54.989 UTC
8	8	2018-02-23 16:26:48.251 UTC	ad7cc8cb-b69f-44f9-a74e-c3b41b444a20	Vehicle	{"type": "license_plate", "information": {"number": "-."}}	2018-11-23 22:39:54.996 UTC
9	9	2018-02-23 16:26:48.397 UTC	8a69e5d7-06dd-4494-81a6-1ff1aa6c76f4	Vehicle	{"type": "license_plate", "information": {"number": ""}}	2018-11-23 22:39:55.018 UTC
10	10	2018-02-23 16:26:48.543 UTC	ff4cce8a-933e-4787-abc4-d5dc53998a38	Vehicle	{"type": "license_plate", "information": {"number": "JXT242"}}	2018-11-23 22:39:55.040 UTC
11	11	2018-02-23 16:26:48.838 UTC	75c62a20-c3b8-4a13-a8b0-b27ee85a5d88	Vehicle	{"type": "license_plate", "information": {"number": "TZD762"}}	2018-11-23 22:39:55.047 UTC
12	12	2018-02-23 16:26:48.988 UTC	c17b9e00-f8fd-4f31-8925-a94944f7e1f0	Vehicle	{"type": "license_plate", "information": {"number": "LMQ359"}}	2018-11-23 22:39:55.055 UTC
13	13	2018-03-23 15:00:07.394 UTC	51def334-2aa3-4c2b-abdb-da2dbc7d93a5	Vehicle	{"type": "license_plate", "information": {"number": ""}}	2018-11-23 22:39:55.063 UTC
14	13	2018-03-23 15:00:07.394 UTC	51def334-2aa3-4c2b-abdb-da2dbc7d93a5	Vehicle	{"type": "license_plate", "information": {"number": ""}}	2018-11-23 22:39:55.063 UTC
15	13	2018-03-23 15:00:07.394 UTC	51def334-2aa3-4c2b-abdb-da2dbc7d93a5	Vehicle	{"type": "license_plate", "information": {"number": ""}}	2018-11-23 22:39:55.063 UTC

Filas por página: 100 1 - 100 de 12890

Selección final y extracción de campos relevantes de la tabla “verifications”

<https://drive.google.com/file/d/1AILnSHdwnQy9eG0fmksK3jPMEeUnbIqc/view?usp=sharing>

• Tabla Trips

trips		
Esquema	Detalles	Vista previa
Nombre del campo	Tipo	Modo
steps	STRING	NULLABLE
gateway_data	STRING	NULLABLE
id	STRING	NULLABLE
_sdc_table_version	INTEGER	NULLABLE
updated_at	TIMESTAMP	NULLABLE
status	STRING	NULLABLE
shipper_id	STRING	NULLABLE
network_id	STRING	NULLABLE
_sdc_received_at	TIMESTAMP	NULLABLE
_sdc_sequence	INTEGER	NULLABLE
amount	NUMERIC	NULLABLE
comments	STRING	NULLABLE
created_at	TIMESTAMP	NULLABLE
gateway	STRING	NULLABLE
_sdc_batched_at	TIMESTAMP	NULLABLE
_sdc_extracted_at	TIMESTAMP	NULLABLE
gateway_id	STRING	NULLABLE

Descripción

Ninguna

Información de la tabla

ID de tabla	nilus-bi:nilus_logistics.trips
Tamaño de la tabla	103,89 MB
Número de filas	13.329
Creación	6 ago 2019 17:47:28
Caducidad de la tabla	Nunca
Última modificación	15 dic 2020 17:33:26
Ubicación de los datos	US

[illegible]

Versión completa de la tabla extraída como CSV en:

Versión completa de la tabla extraída como JSON en:

https://drive.google.com/file/d/1o9nxeQ96pjX2FYlmAl2IUXZ_4wwu85Ul/view?usp=sharing

Tabla “trips sin repetidos tabla” para referencia en lugar de “trips”

trips_sin_repetidos_tabla

Esquema	Detalles	Vista previa
Nombre del campo	Tipo	
steps	STRING	
gateway_data	STRING	
id	STRING	
_sdc_table_version	INTEGER	
updated_at	TIMESTAMP	
status	STRING	
shipper_id	STRING	
network_id	STRING	
_sdc_received_at	TIMESTAMP	
_sdc_sequence	INTEGER	
amount	NUMERIC	
comments	STRING	
created_at	TIMESTAMP	
gateway	STRING	
_sdc_batched_at	TIMESTAMP	
_sdc_extracted_at	TIMESTAMP	
gateway_id	STRING	

[Editar esquema](#)

```
SELECT id, updated_at, steps, status, shipper_id, network_id, amount, comments, created_at
FROM `nilus-bi.nilus_logistics.trips_sin_repetidos_tabla` order by id, created_at
```

[illegible]

